Distributed Mailing System (DMS)

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea "Politehnica" din Timişoara
în domeniul ŞTIINŢA CALCULATOARELOR
de către

Ing. Mezö Patrik Emanuel

Conducător ştiinţific:	prof.univ.dr.ing. Mircea Vlăduţiu.
Referenţi ştiinţifici:	prof.univ.dr.ing. Mircea Stelian Petrescu.
	prof.univ.dr.ing. Daniela Elena Popescu.
	prof.univ.dr.ing. Horia Ciocârlie.

Ziua susţinerii tezei: 23.11.2012

Seriile Teze de doctorat ale UPT sunt:

1. Automatică
2. Chimie
3. Energetică
4. Ingineria Chimică
5. Inginerie Civilă
6. Inginerie Electrică
7. Inginerie Electronică şi Telecomunicaţii
8. Inginerie Industrială
9. Inginerie Mecanică
10. Ştiinţa Calculatoarelor
11. Ştiinţa şi Ingineria Materialelor

Universitatea „Politehnica” din Timişoara a iniţiat seriile de mai sus în scopul diseminării expertizei, cunoştinţelor şi rezultatelor cercetărilor întreprinse în cadrul şcolii doctorale a universităţii. Seriile conţin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susţinute în universitate începând cu 1 octombrie 2006.

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

Cuvânt înainte

Teza de doctorat a fost elaborată pe parcursul activităţii mele în cadrul Departamentului de Calculatoare al Universităţii „Politehnica” din Timişoara.

Mulţumiri deosebite se cuvin conducătorului de doctorat prof.dr.ing. Mircea Vlăduţiu pentru îndrumarea sa şi pentru încurajările din momentele mai puţin favorabile.

Doresc de asemenea sa îi mulţumesc lui dr. Ing. Lucian Prodan pentru sfaturile şi îndrumările sale pe parcursul activităţii mele de cercetare.

Aş dori să le mulţumesc părinţilor pentru suportul moral acordat în timpul celor trei ani de cercetare, susţinere fără de care nu aş fi putut realiza această etapă a vieţii.

Teza de doctorat a fost realizata cu sprijin partial din grantul strategic POSDRU/88/1.5/S/50783, Proiect ID50783 (2009), cofinantat din Fondul Social European "Investeste in oameni", in cadrul Programului Operational Sectorial Dezvoltare Resurse Umane 2007-2013.

Timişoara, Noiembrie 2012 Patrik Emanuel Mezö

Tuturor celor care m-au susţinut.

Mezö, Patrik Emanuel

Distributed Mailing System (DMS)

Teze de doctorat ale UPT, Seria 10, Nr. 41, Editura Politehnica, 2012, 110 pagini, 37 figuri, 4 tabele.

ISSN: 1842-7707

ISBN: 978-606-554-568-7

Cuvinte cheie:
Peer-to-Peer, network, distributed mailing system, mailing architecture, system architecture.

Rezumat, În prezenta lucrare, un nou concept privind infrastructura de corespondenţă electronică va fi prezentat. Vor fi prezentate soluţii la nivel software, arhitectural, protocoalele utilizate în acest domeniu precum şi o altă perspectivă referitor la conceptul de Client / Server pe care majoritatea programelor îl implementează în scopul comunicării in reţea. Sistemul distribuit de corespondenţă electronică nu are un element central care gestionează conexiunile şi bazele de date. Fiecare nod al reţelei (sistem de calcul) îndeplineşte în acelaşi timp partea de Client şi de Server. Obiectivul prezentului document este de a prezenta o cale prin care fiecare sistem de calcul personal poate contribui la o singură aplicaţie devenid astfel parte componentă a acestui sistem.

Table of Contents

List of Figures

List of Tables

Abstract

This PhD thesis describes the research activity carried on as part of the doctoral program entitled "Distributed Mailing System". Several mailing architecture alternatives based on the Peer-to-Peer (P2P) technology are proposed in this thesis to lower the costs of the traditional mailing service available today.

Traditional mailing systems have adopted a server-centric model in handling email traffic over the Internet. Although the traditional mailing providers employ a large number of servers where mail operations are evenly distributed, all the emails are routed to a central gateway, resulting in accessibility issues if the gateway link is severed. Moreover, the necessity of having dedicated buildings and trained personnel for handling large email operations and network traffic is unavoidable.

The Peer-to-Peer concept denotes a virtual network topology above the physical one, where the entire architecture is managed through the application software and sustained by personal computing resources. This model of harnessing resources across the Internet has first gained its popularity through the file sharing applications available even today. The architecture type of such complex systems is developed over hybrid and structured network models. The hybrid model defines a network environment managed through several server-centric elements and the structured model provides a framework for the above running P2P applications.

Peer-to-peer mailing architectures were developed in response to the high costs and numerous issues of using client-server mailing infrastructures. Through this implementation design, every participant to the mailing system has to share some of its computing resources, such as bandwidth, computing power, storage space, etc. But this implementation also has some structural flaws. The most pressing one would be the unpredictability connection status of peers in the network. As a result, most of the current P2P mailing services were developed as independent entities where no interoperability was provided with the traditional email services.

Through my research activity I have been able to solve some of the P2P structural flaws by developing new mailing architectures on both the hybrid and structural network models. The thesis research field includes network topology and scalability, privacy and security, data consistency, interoperability and platform environment. I have built the entire mailing service on the existing network topologies, adding my contributions through extending them accordingly, to provide data consistency and security. By extending the framework of structural P2P network I have provided an environment where several P2P applications can run at the same time using the same computing resources registered as peers in the network. Data caching occurs according to a prediction method by analysing peer behaviour within the network, assuring this way a stable network holder for the email content and user information. I provided the facilities of load balancing by distributing mail tasks among participants according to a resource evaluation method. This enables each peer to effectively contribute to the mailing system according to the real evaluation of their resources, therefore increasing overall application performance and reliability.

Published Papers and Impact

This research work was sustained by the following publications:

- **P. E. Mezo**, M. Vladutiu and L. Prodan, "Design of a Hierarchical based DHT Overlay P2P routing Algorithm", 11th IEEE International Conference on Computer and Information Technology, Paphos, Cyprus, Aug. 2011, pp. 415 – 420, ISBN: 978-1-4577-0383-6 (BDI, IEEE rank).

- **P. E. Mezo**, M. Vladutiu and L. Prodan, "Interoperability solution between Peer-to-Peer and Client-Server based mailing systems", 2011 IEEE 17th International Symposium for Design and Technology in Electronic Packaging (SIITME), Timisoara, Romania, Oct. 2011, pp. 45 – 48, ISBN: 978-1-4577-1276-0. (BDI, IEEE rank).

- **P. E. Mezo**, M. Vladutiu and L. Prodan, "Distributed Mailing System (DMS)", 2011 IEEE 17th International Symposium for Design and Technology in Electronic Packaging (SIITME), Timisoara, Romania, Oct. 2011, pp. 349 – 354, ISBN: 978-1-4577-1277-7. (BDI, IEEE rank).

- **P. E. Mezo**, M. Vladutiu and L. Prodan, "HMail: A hybrid mailing system based on the collaboration between traditional and Peer-to-Peer mailing architectures", 2012 IEEE 7th International Symposium on Applied Intelligence and Informatics (SACI), Timisoara, Romania, May. 2012, pp. 255 – 260, ISBN: 978-1-4673-1014-7. (BDI, IEEE, Australian Research Council list class C rank).

- **P. E. Mezo**, M. Vladutiu, L. Prodan and F. Opritoiu, "DMail: Distributed mailing system based on the collaboration between traditional and Peer-to-Peer mailing architectures", 2012 International Conference on Information Engineering, Lecture Notes In Information Technology, Vol. 25, Singapore, Singapore, Jun. 27-28, pp. 128 -135, ISBN: 978-1-61275-024-8. (Ei Compendex, Cambridge Scientific Abstracts, Google Scholar, IEE, ISI rank).

1. Introduction

> "The two words 'information' and 'communication' are often used interchangeably, but they signify quite different things. Information is giving out; communication is getting through." Sydney J. Harris.

One of the greatest breakthroughs in the digitalized society was the communication possibility facilitated by the Internet technology. At its origin, the Internet was designed as a shared resource among participants [1]. The model of harnessing each individual entity in the network was a much complex task to handle in an environment of its early stages (ARPANET – late 1960's). The goal of this network was to share computing resources over the U.S, integrating different type of networks (universities, laboratories, etc.) into a single infrastructure. This concept had to place the Internet at another perspective, where the participants had to play an equal role in the network. This approach had to include a variety of new elements that must contribute to the environment stability, such as: probability prediction tasks, shared resources management, bandwidth management, etc.

Initially ARPANET was designed only for communication between research institutions, but through the adopted Network Control Protocol (NCP) allowing host-to-host communication [2], users were able to develop new applications by which this network design became very popular. One of the first breakthroughs in developing applications across the network was marked by the electronic message delivery mechanism (*electronic mail*) designed by Ray Tomlinson. In 1972 the mailing application was modified to run across the ARPANET network and the "@" symbol was chosen for the first time.

The ARPANET popularity was stimulated by the widely use of email application. Once the mailing application was available on every of the ARPANET hosts, the network traffic grew significantly. Because the NCP protocol was developed only as a device driver and its founders could not predict the rapid growth of the ARPANET popularity, Vint Cerf and Bob Kahn initiated a new protocol design and the result was the TCP protocol (Transport Control Protocol) published in 1974. In 1978 the TCP was split into TCP and IP and in between 1981-82 the first plans were made to migrate from NCP to TCP, which marked the birth of the Internet available today.

Now, the Internet is a shared resource, a cooperative network built from millions of hosts all over the world. There are millions of applications that use the network, placing strain on the most basic of resources: bandwidth [1].

Initially built for communication, the Internet has become a large information holder. Recently statistics show that its dimensions have doubled every year, having an actual storage capacity of hundreds of Exabyte [2] (1 Exabyte = 1024 petabytes = 1024^2 terabytes). This was possible through new storage technologies, which brought larger capacity and diminished device size. As the mobile technology emerged in the digitalized society, the number of Internet users has grown significantly.

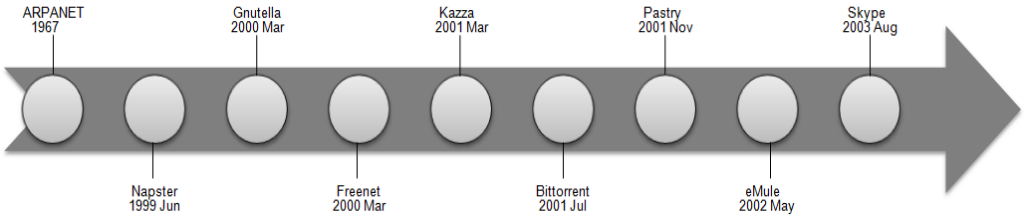


Figure 1.1 Timeline of the Most Popular Peer-to-Peer Applications

With the permanent growth of the Internet resources across all over the world, the network communication standards were also constantly improved. Any participant to the Internet resource can exchange information with other parties through aiming queries precisely at a well known destination addresses, assigned to every computing resource in the network (Internet Protocol (IP) address). The beginning of the year 2011 has marked the exhausting of the current network standard (Version 4) in assigning computing resources with a unique address. Although the event of exhausting IPv4 addresses was a known fact, the process was delayed through mechanisms such as: class-full network design, Network Address Translation (NAT) and Classless Inter-Domain Routing [3]. Because the barriers of 2^{32} addresses space of the IPv4 were exceeded, the developing of its successor IPv6 started at the beginning of 1990's. The Ipv6 has an address space of 2^{128} and its deployment began in the middle of 2012.

Because the model of harnessing every computing resource over the network was a difficult task to handle, the concept of master/slave has gained popularity among the Internet applications. Through this model, a natural segmentation of the Internet was possible. Now every application over the Internet runs according to its own protocol standard, ranging from web based applications to common operating system applications. All the mentioned examples imply the necessity of two kinds of participants: Client and Server. The server usually represents a high end computing resource able to provide services to the requesters in a predefined order. The clients are represented through less capable resource machines that are only able to request the facilities from the server side.

A new concept design that has imposed and encouraged the Client/Server model is available by accessing the facilities of the "Cloud" resources. This solution facilitated the possibility of handling data remotely gaining its popularity by the following diversity of features: availability, security and consistency of data, virtualization possibility of operating systems across the network, etc. Although this solution provides several benefits for the end users, this implementation scales proportional with the costs involved for managing such systems.

A balance was gained through the Peer-to-Peer network concept, which was developed as an opposite model of the Client/Server. Trough this concept design a network architecture model was built virtually above the physical one, generating a second tier of the Internet resources. Every participant to the P2P overlay network adopted both the Client and Server model facilitating and requesting resources at the same time. This new model concept has been imposed by the active society of the Internet and is represented through a select community. Its beginnings were marked by the natural desire of sharing and communication. This concept has first grown in popularity through the applications of file sharing (Figure 1.1), generating

this way other research fields such as content storage platforms, distributed task operations, messaging, etc. In Figure 1.1 some of the most popular applications developed across P2P overlay network are shown through a timeline that marks the beginning of the Internet: Napster [4], Gnutella [5][6], Freenet [7], Kazaa [8] and eMule [9].

The Peer-to-Per concept has a significant contribution in providing incentive mechanisms, stimulating new connections to the world's biggest shared resource available today: the Internet. Although the generated environment was stimulated by the necessity of communication and sharing, the architecture design reflects the natural desire of opening or closing a certain conversation. This principle represents the foundation of such unstable and unpredictable network architectures where participants can join or leave the network at any time. Many of the research areas concentrate on this topic, trying to develop ways in which this unstable environment can be transformed into a predictable one.

The Peer-to-Peer network architecture was designed to harness the available computing resources over the Internet. Compared to the Client/Server model, where the demand of resources are proportionally to the increased number of users, the Peer-to-Peer concept handles resources in a more efficient way. The resource demands do not vary in an alarming way. Although this solution has the lowest cost on the market, it has also its downsides: increased network bandwidth usage, unstable network environment, security and privacy issues.

1.1. Motivation

An important element of our society is communication. Since the beginning of mankind, people have been trying to develop new ways to interact. As society evolved, so did communication skills. A definition of communication, suggest that it is a process of transferring information from a person to another. The tools of communication may involve writing, drawing, sound or gestures.

Nowadays one of the most common communication tools is electronic mail (e-mail or email). Built on a server – centric architecture [10][11], the mailing system relies on two concepts: client and server. An email client is a front-end application that connects to an email server facilitating the operations of reading, sending and deleting email content. The term server describes here a complex architecture, where several entities are grouped together to coordinate processes such as: receiving, storing, replicating and delivery of email content.

Although email tasks are evenly distributed among cluster servers, email traffic is forwarded to a central gateway where email content is processed. There are also scenarios where failures are caused over the traditional mailing architecture design: accessibility issues when the central gateway lies behind an access link that has been severed or flooded, storage stress due to multiple email attachments and server processing stress [10][11]. Another issue arises from the costs supported by the mail providers in terms of dedicated buildings distributed geographically and specialized trained personnel for maintenance and quality of service.

Peer-to-peer mailing architectures were developed in response to the high costs and numerous issues of handling Client/Server mailing infrastructures. Through this implementation design, every participant to the mailing system has to share some of its computing resources, such as bandwidth, computing power, storage space, etc. But this implementation also has some structural flaws: due to

peer member behaviour (peer status is unpredictable – a certain peer can join or leave the network at any time), it is very difficult to handle a complex architecture design like the mailing system, which implies storage space, data availability, bandwidth and computing power.

Two different architectural concepts, structured and unstructured, have attempted to solve this issue in P2P network implementations. The unstructured solution has promoted peers with above average computing resources (Super Nodes - SN) over participants that could/would not share their resources. Mailing systems developed on this concept were designed to rely their backbone on Super Node entities, establishing a reliable and stable network environment.

The structured concept was developed as a P2P framework for other applications. This solution handled peers into a single identifier space and data had to be placed at keys correlated with addresses within the overlay layer. This solution has solved the issues raised by the hybrid model in terms of limitation of queries for a better bandwidth latency usage and any peer could be directly addressed within the identifier space. The mailing architectures developed on such frameworks are more complex in terms of architecture design, security, data availability and overall distribution of tasks.

Considering the overview presentation, I propose new architecture designs, where every personal computing resource contributes to a single application system and becomes a part of it. I entitled my work: "Distributed Mailing System (DMS)", through which a complex mailing architecture design is shaped by combining both the peer behaviour, in terms of time spend over the Internet, and computing resources evaluation methods. The distributed mailing system has no central unit for managing connections and email content. There is no central server designed to serve a certain task. All peers (computing systems) fulfil the Client and Server part, where the whole system resources rely on end user computing systems. Based on the P2P technology, such as [4-9], users have been able to harness their computing resources to a global community. I have adopted the same technology to shape a network architecture design, where attempts to centralize elements within a decentralized system were made.

The domain of this Ph.D. thesis relates to the aspect of designing an e-mail system where the entirely data and communication relies on end user systems and compatibility with other mail systems is maintained.

The direction of research sets the basics for distributed mailing systems, such as proposing a network architecture design for load – balancing data availability in a stable environment. Further, my goal is to build the proposed system within an unstable distributed network environment, as outlined in this thesis.

The proposed thesis enrols under the Distributed Computing domain addressed by the sub - domain of Distributed Computing Architecture domain and Distributed Computing Cluster domain.

1.2. Thesis Goals

The distributed mailing system relies on two concepts: parallel computing and Peer-to-Peer network design. Parallel computing represents a form of computation where large problems can often be divided in smaller ones, which are then solved concurrently (in parallel) [12].

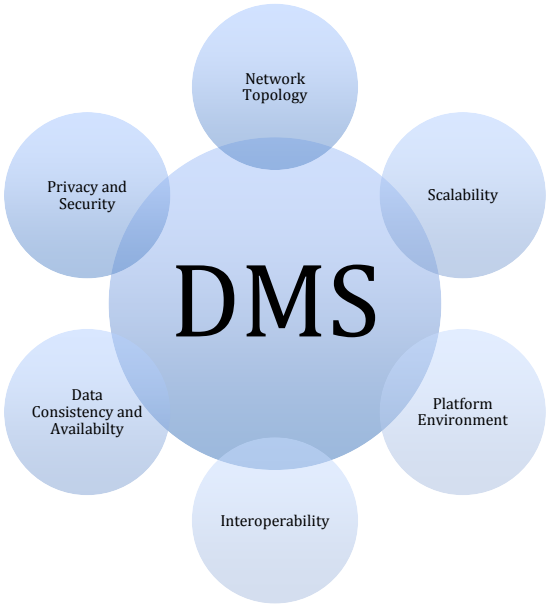


Figure 1.2 DMS Research Fields across Peer-to-Peer Network Environment

Parallelism occurs in many forms such as: bit level, instruction level, data and task level. Parallel computing represents one of the cost effective solutions for processing large and intensive data problems. Data intensive applications are represented through: transaction processing and information retrieval, data mining, analysis and multimedia services. DMS mailing system implements the parallel computing concept at instruction and data level.

Instructions are distributed to other computing systems through network queries and caching email content relies on algorithms designed for parallel architectures.

The thesis main goal resumes in building a stable environment across the P2P virtual network, generating this way a stable and secure holder for the email content and user information. The research fields that are considered for building a distributed mailing system are found in Figure 1.2. Although several mailing architectures are presented in this thesis, every research area considered in Figure 1.2 represents one of the goals for building a stable data holder across the P2P virtual network.

The first goal in building a mailing service across P2P network is finding a network topology suited for adapting the email operations from the traditional service to a distributed one. In this thesis I used existing topologies, but also added my contributions trough extending them accordingly, at a community based environment. Through this segmentation at community level, I can provide load-balance among email tasks and network bandwidth latency usage.

One major issue arises when handling the unstable platform environment generated by the joining peers. Every participant to the mailing system joins the P2P network with an uncertain online status. The second goal of this thesis is to provide an uptime prediction algorithm based on the peers joining behaviour. In this manner the mailing system operations can be sustained by the network backbone formed by nodes with above average uptime status.

The third goal is to assure the data consistency and availability. This goal can be reached through referring only to the nodes that meet higher requirements (above average computing resources) in terms of uptime, bandwidth, computing power and shared space.

The fourth goal is represented through designing a reliable system interface between the Peer-to-Peer mailing system and current client-server based mailing solutions. Several issues are raised by such an interface, one is the traditional P2P mailing systems need to handle internal protocols by biding to a certain RFC standard format. Another issue lies in the way peers are referred to from the outside network. My approach involves separating the RFC standard from the internal communication protocol between peers, thus enabling the interoperability between systems even if the RFC standard is updated.

Privacy and security of both email operations and email content are assured through layering the network topology at community level and by using the facilities of private and public key encryption. My goal is to provide the minimal security cover for the distributed mailing system design. To perform a fully secured communication, one could easily extend the proposed security model by requiring the services of an external certificate authority, which could provide a higher level of security.

The last goal proposed in this thesis is represented through simulating the mailing system across the P2P network environment and determine if such a system model can be sustained from personal computers and if it does scale well under certain case scenarios.

1.3. Thesis outline

For designing an email system that compares or even performs better than the traditional architecture, I had to consider a variety of research domains: network topologies and protocols, Peer-to-Peer architecture structure, current mailing architecture implementation, interoperability issues, security and scalability.

In chapter two I will provide a quick overview regarding the protocol standards used in handling the network communication. The information is provided gradually starting from the standards of developing a certain protocol used in network communication and finishing with my vision regarding the Peer-to-Peer network model.

Chapter three provides a thorough classification and analysis of the current Peer-to-Peer applications and infrastructures. In this chapter I will propose a new P2P infrastructure design that serves as a common platform support for several applications. By this proposal, every application implemented across such a platform type can configure the virtual network according to its desired computing resources. Further, this approach eliminates the issues raised by the differences between Peer-to-peer applications in terms of architectural implementation and network communication.

The issues raised by the unpredictable uptime status of joining peers are also addressed in chapter three. One of the main characteristics of the Peer-to-Peer application implementation is described by the free will of participants to join or leave the network at any time. For this purpose, several P2P application types will be studied, and I will conclude the analysis with designing an algorithm able to predict the moment in time when a certain node is or will be present in the network.

By this approach, I will be able to develop caching techniques of email content across the P2P network environment with a minimum requirement of nodes involved in replicating data content.

In chapter four I will analyse the current mailing architectures. Both the traditional (Client/Server architecture) and Peer-to-Peer based mailing architectures will be analysed. According to the deficiencies and issues raised by the analysed mailing infrastructures I will later base my direction in developing new architectures across the Peer-to-peer environment.

Through chapter five I solved the interoperability issues raised by the incompatibility between the traditional and P2P based mailing architectures. For this matter I have designed an interface model for the feature P2P mailing architectures to adopt in their implementations. I have carefully selected the most popular and used protocol standards in handling the traditional mail communication and the ones used by the Peer-to-Peer model for providing a good foundation of the interface design. I have also considered the intake of resources within the P2P network for providing the support for compatibility between the two mailing models.

In chapter six I was able to harness all my research activity through designing three mailing systems based on the Peer-to-Peer network model, different in terms of architectural implementation, distributed tasks, network topologies and mail operations. My focus remains set on generating first the stable environment across the P2P virtual network and when this goal is reached the email operations can be implemented. Three types of P2P architecture models were adopted for this matter, one of the model relying on the research performed in chapter three. The other two implementations were chosen according to their performance analysed throughout this thesis. The mailing systems are developed according to the interface layout proposed in chapter five, providing a two-way compatibility with the traditional mailing architectures available today, handling outgoing and incoming mails from one implementation to another. I have also provided certain security facilities in handling the mailing operations in terms of sending, receiving and caching the email content. Data consistency and availability is assured by replicating the email content according to a self developed algorithm, able to provide a timeline of peer availability across several days.

In chapter seven I will conclude my work by arguing upon the obtained results and presenting my contributions throughout this thesis.

2. Network Fundamentals

A major breakthrough in the computer society was marked by the communication possibility between random entities. This communication facility has marked the evolution of computing era through constantly improving resources such as hardware, applications, operating systems, transfer mediums, etc.

The term "computer network" [13] describes a collection of connected computers that can exchange data (send and receiving operations) through a shared access medium. Depending on the transmission medium kind (physical cable or RF signal) the network type can be either wired (fixed network) or wireless. Depending on its size, the network can be considered as personal area network (PAN), local area network (LAN), metropolitan area network (MAN) or wide area network (WAN).

The communicating entities in a computer network can be classified as users, hosts and processes [13]:

- The user is represented by a human and all the actions triggered in the network.
- A host is represented by a computing resource identified by a unique ID in the network.
- The process is represented by the application that handles the network operations of sending and receiving data.
    - A server process provides services for the client side.
    - A client process retrieves services from the server side.

The communication between random parties in the network occurs according to a certain defined standard format. Although several protocols were adopted as standard for network communication, every concept design must conform to a common format.

To simplify a complex system, any protocol used in network communications must present itself in a modular way. A layered network model reduces the complexity of problems by dividing it into smaller tasks, allowing standardization of interfaces among network devices and facilitates modular engineering for development at a single layer, without being concerned about what happens at another layer.

Any data exchange among computing resources across the network can occur according to a set of common communication rules. A set of network reference models were developed in order to standardize the way of communication between network parties. Each reference model has a representation of several interconnected protocol layers. The communication takes place gradually, sending the packages through every protocol layer, until the requester protocol has been reached.

2.1. Network Protocol Standards

The International Standards Organization (ISO) Open System interconnection (OSI) Reference Model (Figure 2.1) defines a standardization proposal for the protocols that will be developed. The model was designed as a modular entity proposing a set of seven interconnected layers. Each layer has the task to reduce the complexity of handling network communication from the above layer to the lower one and vice-versa.

The seven layers of the OSI model can be grouped into Host Layers (Application, Presentation, Session and Transport) and Physical Media Layers (Data Link and Physical). The host layers are used in handling communication at higher level between computing resources and the physical layer handles the process of delivering messages across network. The networking functions of each layer is presented as follows according to [14]:

- **Application.** Layer 7. This layer provides an application program interface (API) for the applications that use the network connection. This helps by blending the steps needed for packaging and securing the delivery of data sent over the network.

- **Presentation.** Layer 6. Assures data translation for both the Application and Session layer. When data is to be received for the current host, this layer formats the data according to the computer's own syntax. When data is to be sent from the Application layer, the Session layer formats the data from the host syntax to the common transport syntax.

- **Session.** Layer 5. Assures that two applications can create a persistent communication connection, through establishing, managing and closing connections between processes.

- **Transport.** Layer 4. Includes the mechanism of safely delivering data packets across the network. When data is to be received for the current host, this layer reassembles data packets into a single message for the Session layer. When data is to be transmitted, this layer breaks down the message received from the Session layer into several smaller ones.

- **Network.** Layer 3. Provides logical network routing through sending the assigned packets to their destination paths.

- **Data-Link.** Layer 2. Assures flow control, ordered delivery of frames, error notifications, addressing and network topology.

- **Physical.** Layer 1. The physical transmission environment is established at this layer. The data that is received or transmitted from/to the medium is represented through ones and zeros that are the equivalent of voltage levels used in handling the physical communication.

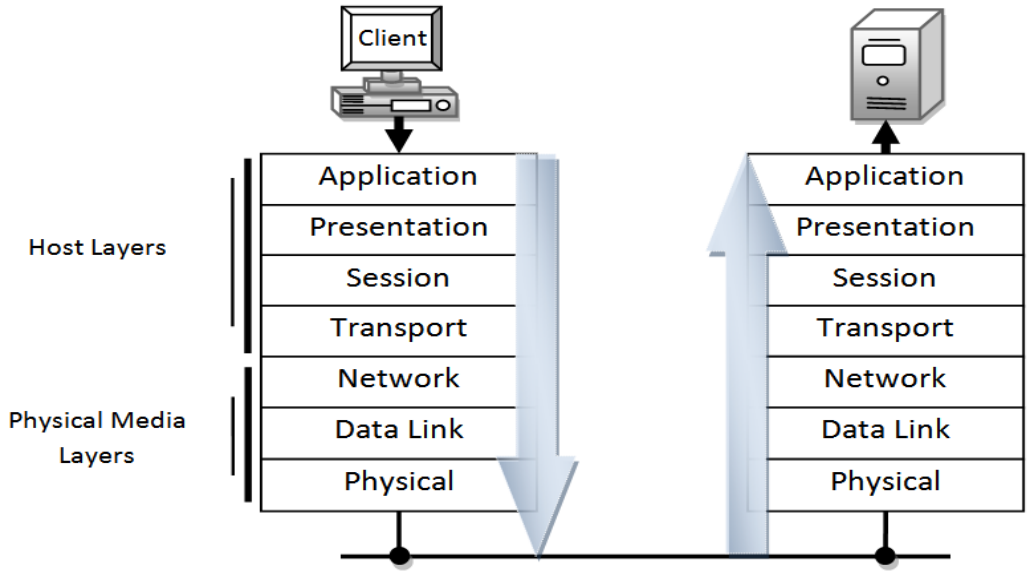


Figure 2.1 OSI Reference Model [14]

Figure 2.1 describes the process [14] of sending data from one host to another (Client-Server) by specifying every layer of the OSI model involved in completing the communication task. When the data is to be received from the Client side, the Application layer passes the data to the Presentation layer. At this layer level, data formatting occurs into the transport layer syntax. When the Session layer confirms that the destination host is ready to receive data, the Transport layer is notified. The transport layer breaks down the data into smaller packets labelled so that the message can be reassembled again. Every data packet is appended with a destination header at the Network layer. The header consists of source and destination logical address. At the Data-Link layer level, frames are attached to the data packets, consisting in error checking, data offset, etc. The lowest layer involved in securely transmitting data over the network, Physical layer, effectively transmits the data packets at bit level trough the network transport medium (wired or wireless medium).

When data is to be received at the server side, a similar process of reassembling data through the arrived packets from the network medium takes place.

2.2. Transmission Control Protocol / Internet Protocol

The TCP/IP protocol represents the research result of the first network protocol attempt, ARPANET, founded by the Defence Advanced Research Projects Agency (DARPA) [2]. The development of this protocol was critical in allowing the future growth and scalability of the largest network architecture available today: the Internet. The TCP/IP protocol provides a connection oriented binding between two entities, guaranteeing that every data packet that is sent in the network to its destination is successfully received.

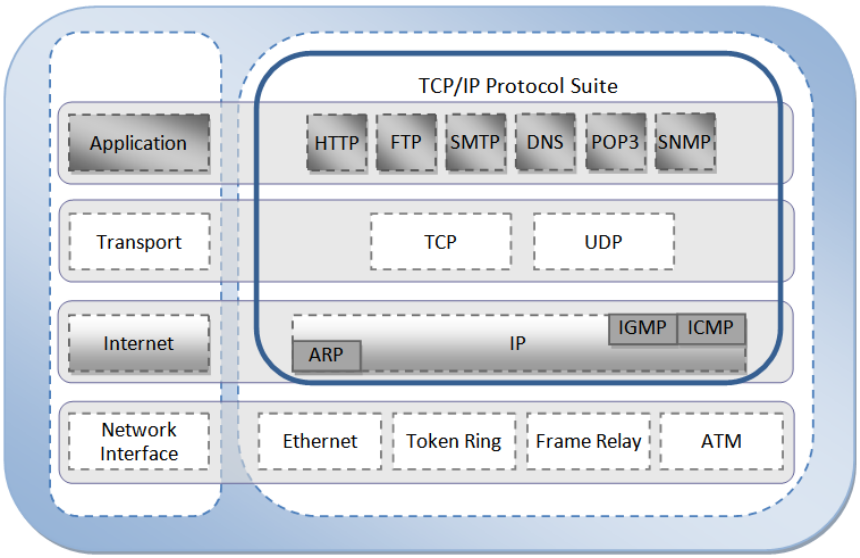


Figure 2.2 TCP/IP Protocol Suite [14]

A set of four layered protocols form the TCP/IP model (Figure 2.2): Application, Transport, Internet and Internet Interface. The four protocols are handled as independent tasks within the TCP/IP stack [14], generating this way several benefits:

- Multi-platform compatibility is generated trough the facility of having several protocols developed at the same layer level.
- Applications can require specific services provided by a protocol within certain layer level.
- Development of various protocols can be implemented at any of the four layers simultaneously due to the layered design of the TCP/IP model.

The TCP/IP relates to the OSI model by structuring the layers accordingly:

- Application encapsulates the Application, Presentation and Session OSI layers.
- Transport has the same functionality as the OSI layer.
- OSI Network layer is handled by the Internet TCP/IP layer.
- Data-Link and Physical are represented through the Network Interface.

The Application layer of the TCP/IP model facilitates the communication between applications and the network resources. Although this layer provides the support of several protocols running simultaneously, it provides an additional

feature by running different applications using the same protocol. This is possible through assigning each connection successfully established, with the other network parties, with a socket specified by the network address and the port number of the connection in progress.

Throughout the evolution of computing system resources stimulated by software development, some of the protocols adopted by applications mainly for network communication have reached a standard state in the Application layer of the TCP/IP model (Table 1 [14]).

Table 2.1 Standard Applications Protocols [14]

Protocol	*Description*
HTTP	Hypertext Transfer Protocol. Designates the protocol between Web browsers and Web servers.
FTP	File Transfer Protocol. Performs file management between remote computers.
SMTP	Simple Mail Transfer Protocol. Protocol used for email delivery between mail servers.
DNS	Domain Name System. Assigns hostnames to IP addresses.
POP3	Post Office Protocol version 3. Used by mail clients to retrieve the email content.
SNMP	Simple Network Management Protocol. Used for gathering information about network devices which is define in the Management Information Base (MIB).

Table 2.1 provides a hierarchical layering of protocols used for mail process inter-communication. The TCP/IP model represents the base layer where other mail exchange application protocols are built. The Simple Mail Transfer Protocol is used for inter-server communication, so that the email message can travel across several servers until it reaches destination. For ensuring the correct location addressing, the implementation of Domain Names and Dynamic Host Control protocols are required. Multipurpose Internet Mail Extension (MIME) refers to the mail format, extending it by enabling other character sets than ASCII, multiple attachments or message bodies with multiple parts. Both Post Office Protocol (POP) and Internet Mail Transfer protocols are developed for the receiving process of email content. The most basic retrieving operations are implemented through POP protocol and more complex facilities are obtained by implementing the IMAP across mail servers.

The Transport layer of the TCP/IP model guarantees a reliable connection between host to host transfers. A connection is represented through a logical association between entities from different systems [3]. Most of the data transfers over the network occur by implementing the TCP layer (Transport Control Protocol). The TCP protocol guarantees a reliable network connection through confirming that all the data packets are successfully sent to their destination.

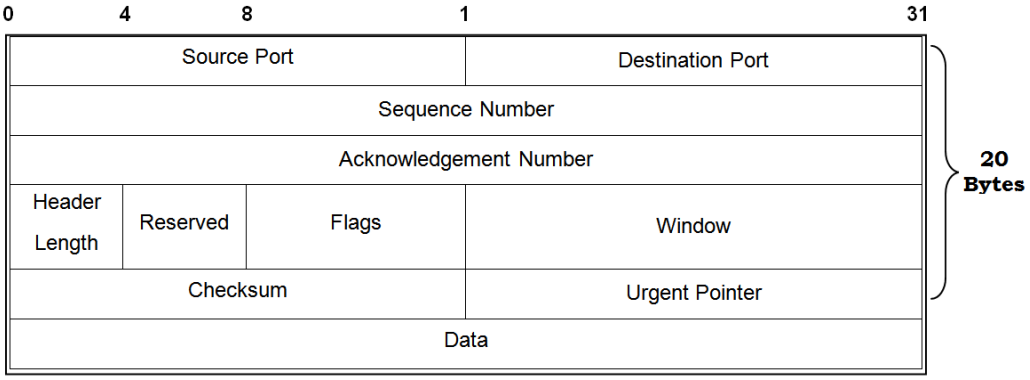


Figure 2.3 TCP/Header [3]

Figure 2.3 shows detailed aspects of the TCP header. Both source and destination ports are 16 Bit long, and represent a unique identifier of the requester or targeted process (application). Sequence and Acknowledgement numbers provide the flow control. Because network packets do not always arrive in a predefined order (routed through different paths or may be dropped), the sequence number is used to reassemble data and also makes requests for the lost packets. Checksum is used to detect errors in the TCP segment. Packets that fail checksum get retransmitted.

Table 2.2 Internet Layer Description [14]

Protocol	*Description*
IP	Internet Protocol. Provides data routing and addressing of data packets within the network.
ARP	Address Resolution Protocol. Hardware addresses of hosts within local network are obtained through this protocol.
IGMP	Internet Group Management Protocol. Manages host membership and IP multicast group.
ICMP	Internet Control Message Protocol. Handles error reports regarding delivery of data packets.

UDP (User Datagram Protocol) provides a connectionless and unreliable communication [14]. Applications relying on this protocol are also responsible of safely delivering data packets over the network. Because there is no waiting involved for confirmation of the sent packets, the UDP protocol provides faster communication than TCP. This protocol's orientation purpose is based more on the audio/video streaming applications.

The OSI Network layer is represented through the IP protocol in the TCP/IP model. IP is used for routing purposes of the data packets within the network. Four layers form the IP layer, presented in Table 2.2.

The Network Interface layer forms the Data-Link and Physical layers of the OSI model. This layer handles the transmission of data trough the assigned transport medium of the network.

As presented in this chapter, the TCP/IP model is based on a suite of 4 layers, each one designed to work independently from the other layers. Although many protocols were developed through developing applications that require network access, only few were adopted as standards in the TCP/IP model. Every protocol that has reached a mature state can be proposed as a standard by publishing its implementation in a series of documents called requests for comments (RFCs).

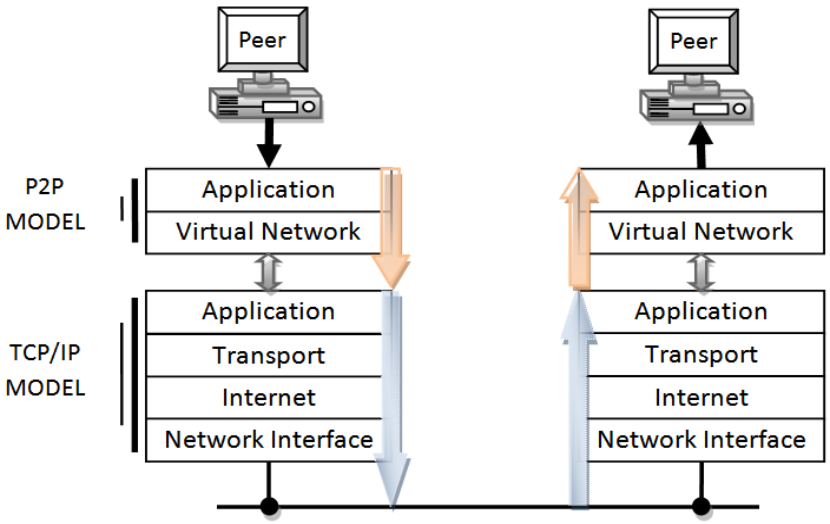


Figure 2.4 Peer-to-Peer concept related to OSI Model

2.3. Peer-to-Peer towards TCP/IP model

The Peer-to-Peer model concept designates a different approach in handling hosts over the network. Although in its early stages, the Internet was built on a point-to-point communication model, the Client/Server concept has gained popularity among applications that want to use the resources of the largest network available today – the Internet. The Peer-to-Peer concept combines the Client/Server model at the same host side. Hence, a peer communicating with other peers in the network can implement both client and server properties.

The Peer-to-Peer concept does not designate a new breakthrough in the digitalized society. Its early stages began with the applications of file sharing [4-9], most of which are available even today in an active state of continuous development. As many applications have been developed, so did the protocols implementations needed for the peer-to-peer communication.

Every Peer-to-Peer concept denotes also a multitude of computing domains such as: network topology, bandwidth, uptime, computing resources, etc. When handling the network topology, a protocol is built over an existing standard one, which usually is represented through TCP/IP. Hence, the Peer-to-Peer applications

reconstruct a model related to OSI above the application layer (Figure 2.4). In order to standardize the way that peers communicate over the network, overlay models have been developed [15-19]. The overlay concept tends to standardize a protocol through providing an interface for the P2P applications that want to connect to the network.

Distributed Hash Table (DHT) based overlay networks provide the framework support for P2P applications. The mechanism behind the virtual network is self-organizing through the operations that facilitate scalability, load-balance, decentralization and availability. Through the Hash table, an identifier space is created similar with the IP address from the TCP/IP model, hence virtual IP (Figure 2.4). The virtual Network corresponds with the used topology in handling peers over the network (ring, mesh, etc.).

When designing a Distributed Mailing System, we have considered the TCP/IP model as the fundamental protocol, which lies beneath the P2P protocol used for the inter-peer communication. Although in this thesis we present several mailing systems, every design has its own network protocol implementation. We will not present the protocols used in this thesis, we will refer only to the used topologies and provide interfaces to the standard protocols. For this matter we provide an interoperability solution with the mailing services available today.

3. Peer-to-Peer

The Peer-to-Peer concept denotes a network architecture model above the physical network structure. The participants that architect the system are called peers and in most cases they are represented by personal computers that share resources such as computing power, bandwidth and storage space. The P2P concept was first introduced in file sharing applications, continuing its presence afterwards in other fields such as: voice over IP (VOIP), mailing systems, social applications, etc.

The participants that contribute to the P2P network architecture are treated as individual computing resources that share a common characteristic: at the application level, a virtual network is shaped according to its own routing mechanism. The topology used in achieving the virtual network above the physical network layer has a significant influence on the system (application) performance, reliability and in some cases anonymity. The virtual topology has also a significant influence in terms of bandwidth costs: some P2P implementations communicate through broadcast messages and others aim messages directly to the requested destination.

The Peer-to-Peer infrastructure defines no standard implementation, it was shaped under the circumstances of developing new applications that facilitate operations in a distributed environment. In this context, the authors of [6] have identified several requirements regarding the implementations of P2P architectures:

- **Ability to operate in a dynamic environment***:* the P2P network environment permanently changes due to peer member behaviour (unable to predict when a peer is connected to the network). Applications should achieve transparency in handling data availability, security and anonymity in the P2P network.

- **Performance and Scalability***:* ideally, when scaling the network, the storage space and data availability should grow linearly and the response time should remain constant.

- **Reliability***:* failures over the network should not cause significant data or performance loss.

- **Anonymity***:* is achieved in terms of privacy regarding the search queries, unpopular information and peer (host) identification.

Another attempt to characterize the P2P environment was concluded in the following definition [20]:

"*Peer-to-Peer systems are distributed systems consisting of interconnected nodes able to self-organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server or authority"*.

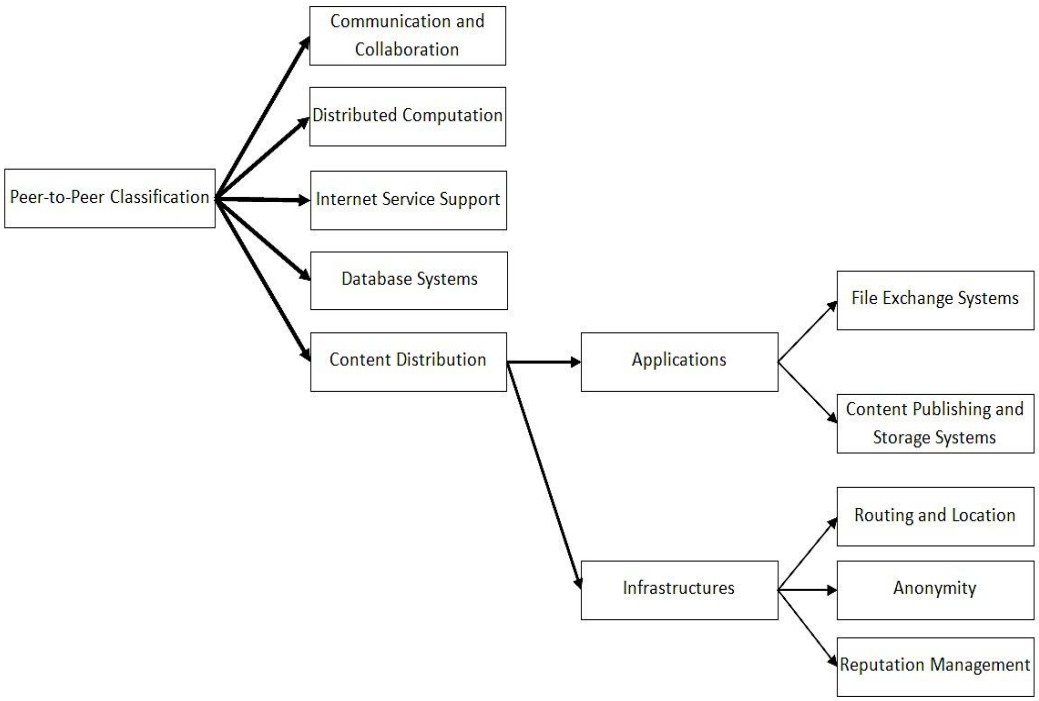


Figure 3.1 Peer-to-Peer Application Classification [20]

By this definition, the authors [20] referred to the P2P concept as abstract as possible by outlining every strong characteristic of this research area. The authors provide also an integration model for the systems with different "degree of centralization", including applications that are fully decentralized (Gnutella [5]) and partially decentralized (Kazaa [8]).

3.1. P2P Applications – Case Study

Throughout the evolution of Peer-to-Peer applications, a range of architecture designs and implementations were also developed. The work presented in [20] provides a good foundation for Peer-to-Peer application classification:

- **Communication and Collaboration.** The systems that facilitate real-time communication are included in this category, such as instant messaging (Yahoo messaging) or VOIP (Skype [21]).

- **Distributed Computation.** Systems that share processing power are included in this category. By this approach, tasks that require intensive workload can be split into several small ones that are sent to the corresponding peers for processing. A central coordination is required for handling the task distribution and result collection and processing. An

example of such systems is represented through the projects of SETI [22] and GenomeAtHome [23].

- **Internet Service Support.** Peer-to-Peer applications that provide Internet services facilities are included in this category. Applications such as Internet indirection (Chord [19]) and security services (SOS [24]) are considered for this purpose.

- **Database Systems.** One of the major challenges faced by the Peer-to-Peer architecture design was handling the shared information among peers. Such systems designs had to handle facilities such as data availability, consistency, security and enhanced indexing. Such applications that focus mainly on handling data over the P2P network can be found in [25], which suggests that all data can be comprised of inconsistent local relational databases, and [26] which describes PIER – a network topology that provides relational queries in handling data search engines.

- **Content Distribution.** Most of the available Peer-to-Peer applications are included in this category. Through this classification, current systems are organized into application oriented concepts and virtual network infrastructures. The systems that provide the application sharing facilities are mainly focused on file exchange and content publishing operations. A distinction between these two can be made through the generated shared platform: file exchange applications [5-9] concentrate their resources mainly on searching and transferring files between peers and content publishing applications are more focused on generating the way through which peers can publish, store and distribute content across the network (Oceanstore [27]). The second subcategory of content distribution systems is represented by the infrastructure framework providers. This provides P2P applications with a predefined API, which facilitate operations to easily implement across the virtual network. The routing and location services provide an efficient environment for addressing queries. The addressing space is related with the information that will be stored. Here systems are mentioned such as CAN [15], Pastry [16], Tapestry [18] and Chord [19]. A second subdivision of the infrastructure providing systems is oriented on providing user anonymity. Systems such as Freenet [7] and Onion Routing [28] provide confidentiality among peers. The Reputation Management subdivision provides a central organization to maintain information for users and their behaviour – PeerTrust [29].

Throughout this thorough analysis of current Peer-to-Peer systems, the authors from [20] identified several attributes that highlight the strengths of certain concepts. Figure 3.1 is self explanatory through illustrating the design decisions that have a direct impact on the resulting attributes:

- **Security**. Further analysed in terms of Integrity and Authenticity. A document cannot be modified or substituted by unauthorized entities.

- **Privacy and Confidentiality**. Data is only available to those authorized and a control of how information is collected and used is provided.

- **Availability and Persistence**. Ensuring that data is available to authorized users when required.

- **Scalability**. An increased number of nodes should not affect the performance and availability of P2P systems.

- **Performance**. Reduced time performance in handling operations such as publication, searching and document retrieval.

- **Fairness**. Ensures that users have access to resources in a fair and balanced manner.

- **Resource Management and Grouping**. Operations such as publishing, searching, content retrieval, editing/removing documents and management of storage space are provided.

- **Semantic Grouping of Information**. Content distributed systems are closely to a community related to the peers that share common interests.

The decisions made throughout developing a Peer-to-Peer content distribution system are crucial when it comes to highlighting the attributes that have a great impact on the efficiency of the architecture design. In the next part I will focus mainly on the structural decisions [20] through which P2P applications and infrastructures are developed.

The *distributed location and routing* of data within Peer-to-Peer networks has a great impact on overall performance and efficiency of such architecture designs. From the centralization point of view, P2P systems can be implemented in a purely, partially or hybrid decentralized manner. Purely decentralized systems (Gnutella [5][6]) do not require any coordination from a server-centric element and every node implements the same amount of tasks as the others. In a partially decentralized system, nodes are treated distinctively according to their resources intake to the P2P network (Kazaa [8]). A super node is represented through a computing system with above average resources (increased bandwidth, uptime, processing power, etc.) and its purpose is to coordinate other nodes that do not meet the same amount of resource requirements. A hybrid decentralized P2P system requires the coordination of a server-centric authority whose scope is to facilitate the interaction between peers. Applications such as Skype [21] meet such requirements, where users at the login state, firstly connect to a server for authentication, and if succeeded, the binding to the P2P network occurs after.

From the network structure point of view, Peer-to-Peer architectures can be described as unstructured, structured infrastructures or structured systems. The unstructured applications provide only the location and search mechanism operations within the P2P network (ex. Gnutella [5][6]). The data available for peers to manipulate is not related to the application architecture and search mechanisms are limited according to a TTL (time to leave) descriptor and usually flood the network. The structured infrastructures (ex. Chord[19]) hold both data and routing mechanisms related through the distributed hash table (DHT). The structured systems provide solutions for exact-match queries (Oceanstore [27]).

Data availability and consistency is assured through *content caching, replication and migration* algorithms. Most of the content distributed systems gain

their popularity according to the variety of choice regarding the shared data. To prevent data loss or data inconsistency, the P2P systems were developed through handling several methods of data replications: passive, caching or migration. Passive replication occurs on peer request of a certain data to be copied. Cache based replication takes place when a data usually is queried through several peers. Through this method, every peer involved in passing the information to the requester, maintains a copy of the data for increasing its availability. Migrating replication is used for increasing the data locality and availability throughout the P2P network.

The subject of data *security* is very important when handling information on content distributed systems. When data is made available on the Peer-to-Peer network, certain levels of privacy, confidentiality, integrity and authenticity are required. In general terms, security within content distributed systems addresses the *storage, routing, access control, authentication and identity management*. Secure storage refers to the cryptographic algorithms and protocols used in handling data storing and publication throughout P2P network. Through secure routing the problem of malicious nodes attempting to corrupt, delete or to deny access is addressed and solved. The issues raised by access control, authentication and identity management are usually ignored within content distributed systems. This can generate an environment where peers can join a network with multiple identities, posing a threat to systems that employ content replication or fragmentation. Although this issue was several times argued by many implementations to be unnecessary, this problem remains open in terms of the intellectual property management and right management issues.

The *anonymity* decision remains with certain the most encouraged feature of content distributed systems. The shared data content available on Peer-to-Peer networks (ex. Freenet [7]) can be susceptible on targeting privacy, confidentiality and censorship. According to [1], anonymity can refer to: the author of the content, the identity and content of the document itself and the details of a query for retrieval of the content. The *deniability* decision is close related to the anonymity. Through this decision content distributed systems are encouraged to deliberately deny the knowledge of content stored on any of the peers. As a consequence, users cannot be held responsible for the content they store/share within the P2P network.

Content distributed systems evolved through promoting implementations that facilitate the sharing concept among participants. Throughout their evolution, every system provides some *incentive mechanism* to attract peers voluntarily by joining and sharing their own resources. Although resources are "free" for everyone, some peers contribute to the P2P network with their own resources and some are labelled only as consumers. This generates the need of accountability of peer behaviour and its actions. Reputation mechanisms are also required for inspiring peers trust to contribute to the systems. This mechanism can be implemented through several features: associating user comments with the shared resource or implement algorithms that trace the peer behaviour in time.

Any of the content distributed systems provides at a certain level some *resource management capabilities.* The resources that most P2P applications share across the network are concluded in content (files), storage (disk space), computing power and transmission capacity (bandwidth). Some implementations add some additional facilities to the resource management such as removing or updating content, maintaining previous versions of content, managing storage and setting bandwidth limits.

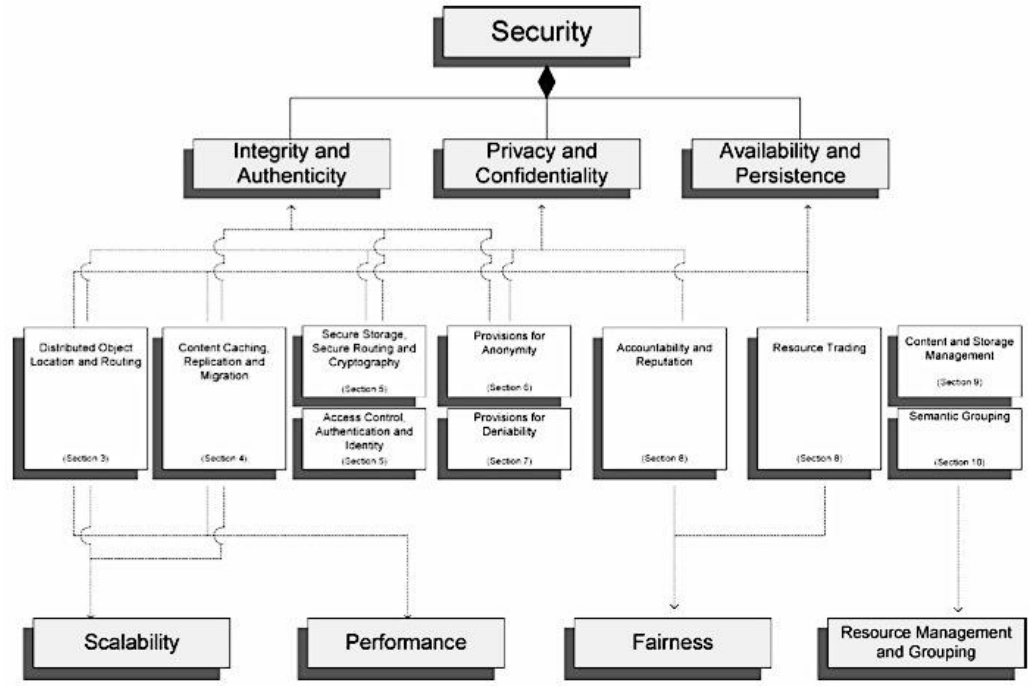


Figure 3.2 Characteristics of Peer-to-Peer content distribution systems [20]

To enhance Peer-to-Peer operations facilities, a notion of semantic grouping of information was developed. Through this approach, content distributed systems are closely to a community related to the peers that share common interests. Throughout this decision, P2P implementations are focusing more on handling operations in a limited manner to the community established through some form of interests. Also this generates reduced costs in handling distributed resources across the network.

3.2. Addressing Scalability in P2P Implementations

Through the provided analysis of the previous subchapter, a classification of Peer-to-Peer architecture designs was established. Although every implementation is unique through combining several techniques for achieving a purely decentralized architecture, the main issue arises when scaling such systems. I will focus mainly on the unstructured and structured infrastructures and provide information about the architecture implementation and protocols used (table 3.1). I have carefully selected few of the most popular applications within the content distributed systems classification. Through the provided analysis of the considered systems, I can select the best performing architectures for developing a distributed mailing system.

Table 3.1 Scalable Peer-to-Peer Architectures

	Centralization		
	Hybrid	**Partial**	**None**
Unstructured	**Skype**	**Kazaa**	**Gnutella**
Structured Infrastructures	**-**	**-**	**Chord**

3.2.1. Gnutella Architecture Overview

Gnutella represents one of the earliest attempts of handling hosts in a fully decentralized manner, where all nodes are performing symmetric tasks, fulfilling the roles of both client and server at the same time. The protocol used in handling peers over the network is developed on top of the TCP/IP model, providing extensions for peers to interact: establishing connection, resource information query and exchange. Gnutella was rapidly adopted by the "peer society" due to the simple protocol implementation and the variety of information that was to be exchanged.

The Gnutella protocol (Figure 3.3) consists of several queries and answers addressed and received from the requester node. Throughout every query/response a descriptor is attached to the information that is sent or received [5]:

- **Ping**. Used for discovering hosts within the P2P network. The receiver usually responds with one or more Pong descriptors.
- **Pong**. Response descriptor to the Ping request. The address of the receiver (IP) and information regarding the shared data are sent along with the Pong response.
- **Query**. Describes the mechanism for searching data within the distributed network. When a match takes pace, the QueryHit descriptor is sent.
- **QueryHit**. Response to a Query. Some information is sent along with this descriptor, to ensure that the queried information can be safely acquired from the receiver side.
- **Push**. A mechanism that allows peers behind a firewall or NAT (network address translation) to contribute to the P2P application.

Through the specified protocol, Gnutella uses broadcast type queries, which implies poor bandwidth latency usage. To overcome query duplicity, nodes store requests and answers when no overload occurs on the current peer. Another solution that overcomes the network flooding consists in assigning every sent query over the P2P overlay network in a TTL descriptor (Time to Leave). Through this method, every time a query reaches another node in the network, the assigned TTL descriptor will be decremented. The forwarding process stops when the descriptor reaches the zero value. To prevent bypassing this limitation, Gnutella protocol also

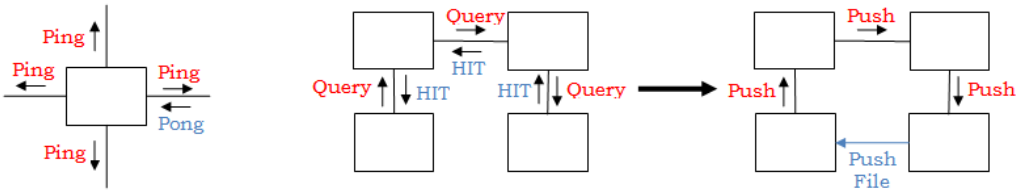


Figure 3.3 Gnutella Protocol - Ping/Pong - Query/QueryHit/Push Routing [4]

specifies the need of a second descriptor – Hop count, which is sent along with the TTL. The query is valid and can pass through if no match was found when:

$$TTL(0) = TTL(i) + Hops(i),$$

- Where TTL(i) and Hops(i) represent the value of TTL and HPS fields of the header of the descriptor's ith hop, i>=0.

By this protocol description I can make the following statement that under such circumstances Gnutella cannot perform well when scaling this architecture design at very large number of nodes. Later versions of Gnutella implementation promote nodes with above average resources with the status of super node. Through this approach, ordinary nodes can connect only to the super node and queries are addressed at this layer only. This represents a major enhancement to the actual protocol of Gnutella, but the innovation represents only another limit to break when it comes to scale this system at a higher number of nodes.

3.2.2. Kazaa Architecture Overview

Kazaa represents the next generation of the Gnutella implementation. Although the protocol never went public, the work in [8] provides a solid foundation of this application architecture design. This system concept is worth mentioning in this subchapter because of its capability in handling an increased number of peers (3 million daily [8]) and a large amount of data available on the overlay (5,000 terabytes [8]).

The architecture design is implemented through extending Gnutella by Gnutella. The overlay is divided into a two tier overlay, the first tier handled by the super nodes and the other from ordinary nodes. The Kazaa implementation exploits the peer heterogeneity by promoting nodes with above average uptime, bandwidth connectivity and CPU power. By this decision, Kazaa handles two kind of operations: super (SN) and ordinary (ON) node tasks. The super node handles connections with other super nodes and ordinary nodes. By handling connections with others of its kind, a super node keeps track for other possible connections by always updating its current list with newly ones queried from its neighbours.

The ordinary node (ON) performs the same operations of changing super nodes addresses obtained from its neighbours. The information that an ON stores on the SN side consists in: the file name, file size, content hash and the file descriptors (ex. author name, album, etc.). The Kazaa application hashes every file to a hash signature, used for identifying and downloading the file content related to the

generated hash signature. Also when the download fails, Kazaa automatically performs the request of another file matching the previously obtained hash for the file content to download.
When query operations take place from the ON to SN, a limitation is set through decrementing a TTL descriptor. This limit prevents network flooding and bandwidth congestions.

Kazaa performs well when handling data in a dynamic environment. The architecture design is focused more on indexing data and providing accessibility range of hashed metadata. The availability is handled by the user's ability to permanently upload new data on the overlay network and by the increased uptime determined by the incentive mechanism of the Kazaa application. Through this approach, Kazaa manages a large amount of data that permanently refreshes due to newly uploaded data. As a conclusion, Kazaa architecture is entirely sustained by the incentive mechanism offered by this application and data consistency and availability are handled poorly through this concept design.

3.2.3. Skype Architecture overview

Skype [21] is included in the category of communication and collaboration and provides facilities such as VOIP (voice over IP), instant messaging, conferencing, avoiding NAT and firewalls, codecs**,** media transfer and buddy lists. Skype was developed by the Kazaa organization, having an architecture design related to Kazaa [8], and moreover, it provides extensions by handling some of the information on top of a third tier – a server centric element. Although Skype architecture implementation and design was not released for publishing, I will use the analysis provided by the authors in [21].

Skype architecture design (Figure 3.4) relies on a three tier overlay composed of the ordinary node (ON), super node (SN) and login server (LNS). The protocols used in handling communications are TCP, UDP and HTTP. Like Kazaa implementation, nodes are carefully selected within the P2P network for achieving the super node state through having increased bandwidth, CPU power and uptime. At this tier layer, connections with other SNs are established in a variant and periodical manner. The links with other SN's are used for handling offline information (messages) or connectivity related issues (bypassing NAT or firewalls).

An ordinary node performs only the operations of end-to-end call or instant messaging. Any ON uses the connections established with several SNs to perform the mentioned operations. The third tier layer is represented through the login server used to store information about user IDs and buddy lists. When a node joins the Skype overlay, it connects first to a known SN and after then performs the authentication process through connecting to the login server. By storing user information on the login server, Skype ensures that no duplicity can occur in terms of user ID and buddy list information.

Additional server centric elements used for PC-to-PSTN and PSTN-to-PC (Public Switch Telephone Network) bridging are implemented through SkypeOut [30] and SkypeIn [31] servers.

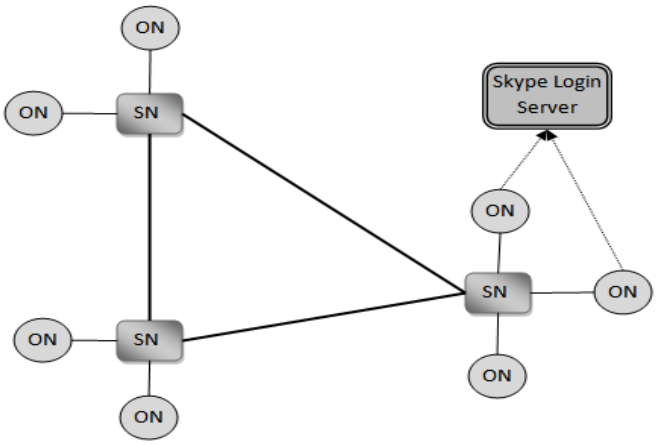


Figure 3.4 Skype Architecture Implementation

An in dept analysis was presented in [21], which provides a more detailed aspect of Skype functionalities:

- **Ports**. There is no TCP or UDP default port on which Skype listens to. Skype uses a range of ports chosen randomly and publishes them (ON/SN) after several connections were established with other Skype clients.

- **Host Cache**. Represents a list of IP addresses and port pairs cached on the local host that periodically is built and refreshed from the Skype Client side. When no connection can be established through the host cache, the node uses the bootstrap IP addresses provided as default in the Skype application.

- **Codecs**. Skype uses the codecs from GlobalIPSound [32], and the frequency range used lies between 50-8,000 Hz.

- **Buddy List**. The buddy list is stored on both local host and login server.

- **Encryption**. Encryption is provided by using AES [33] (Advanced Encryption Standard), also known as Rijndael. Skype uses a 1024 bit RSA to negotiate symmetric AES keys and the user public keys are certified by the Skype login server.

- **NAT and Firewall**. Skype application uses a variation of the STUN [34] and TUN [35] protocols to determine the kind of NAT or firewall a client may lie behind.

Skype represents the next generation of the Kazaa implementation. Through handling connections on top of a three tiered network design, Skype performs well when scaling the system at a large amount of users. Skype's performance comes with a cost: a server centric element manages most of Skype's data availability and consistency. Through analysing this architecture design, I can conclude that hybrid

Peer-to-Peer network implementations represent a solution to the traditional Client-Server model, where information can be handled two sided: the critical data can be stored on the server side and data that is to be "refreshed" (consumed) in time can be stored on the Peer-to-Peer overlay.

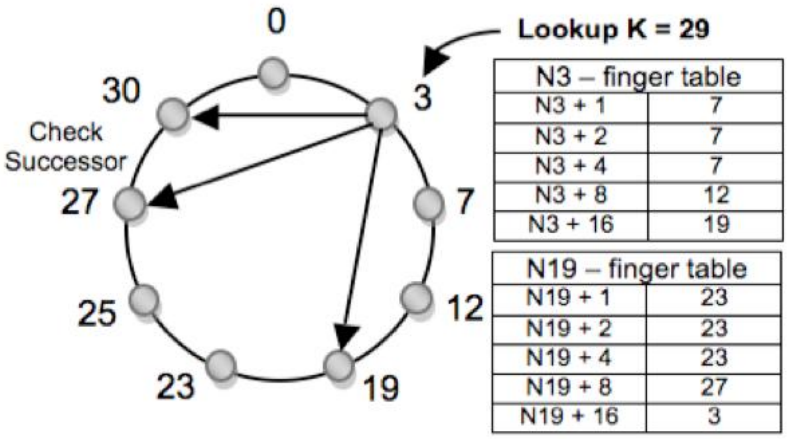


Figure 3.5 Chord Architecture Design

3.2.4. Chord Architecture Overview

With the demand of scaling at a large number of peers, the previous mentioned architectures presented some structure flaws: one lookup query was limited by a time-to-leave descriptor (Gnutella [5], Kazaa [8]) and the entire peer community could not be mapped onto a single identifier space or reached from any point of access in the architecture model. As a response the following designs were developed to overcome these flaws and improve on existing features. I mention here CAN [15], Pastry [16][17], Tapestry [18] and Chord [19].

Distributed Hash Table (DHT) based networks provide the framework support for P2P applications. The mechanism behind the virtual network is self-organizing through the operations that facilitate scalability, load-balance, decentralization and availability. Chord [19] is built in this manner, describing a ring-like virtual network (Figure 3.5), where each node has an unique ID in a m-bit space using the SHA-1 hash function. Every node is linked to its successor and maintains a list of nodes following it in the ring (predecessors). Data to be stored is hashed in the same identifier space as the joining nodes under a certain key k, and is to be placed at the node whose identifier equals or follows k. The node that is responsible for the obtained hash key is called successor(k).

To accelerate the routing process, every node maintains a neighbor table with m entries called *finger table*. The i^{th} entry in the finger table at a node n contains the address of successor($n+2^{i-1}$). Figure 3.5 exemplifies the lookup operation of key 29 from node 3. Node 3 contacts node 19 from its finger table and finds out that the searched key is closer to node 27. When contacting node 27, node 3 finds out that key 29 is stored under the node with the given ID of 30, the successor of node 27.

The Chord implementation represents one of the early breakthroughs in handling Peer-to-Peer network concepts separately from the application layer. As a practical comparison, Chord is somehow related with Gnutella at its beginnings. Although it scales well under any churn conditions, the issues of data availability and consistency are affected.

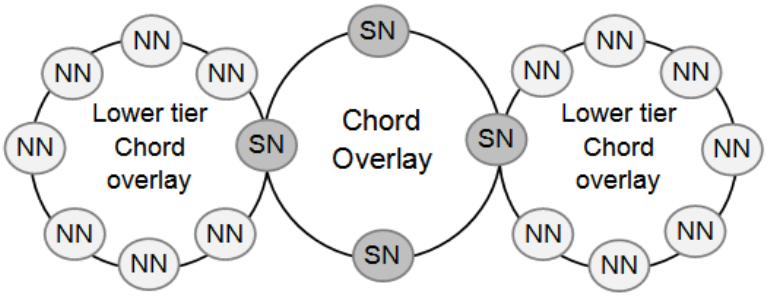


Figure 3.6 Two Tier Chord Overlay Extension

Even if the whole peer community was mapped onto a single identifier space and queries were precisely addressed at any point of architecture map, there is still need to highlight peers with certain properties into different architecture location while maintaining interconnection lookup to a minimum possible. A good example of harnessing the benefits of computing resources within the P2P network is present in [36], figure 3.6.

The two tier extension [36] of the Chord protocol carefully filters nodes with above average resources (*super nodes*) from the normal nodes. The chosen extension describes a structure formed through linking ring topologies to a single one composed of super nodes. The protocol extension resumes in extending Chord by Chord, where every normal node performs the operations described in [17] and the super node duplicates these operations because of its double identity: maintains connections with other super nodes within the first tier and also holds connections with normal nodes from the lower tier. The identifier space of one lower tier lies between the super node and the super node's predecessor ID. The whole mechanism behind this extension is dynamically implemented as when one normal node gains the property of super node, first ask the leading super node for acceptance, and when granted, it can join the first tier having set as lower tier the normal nodes that previously were set as predecessors.

There are several hierarchical implementations that focus on the necessary extensions to fit the demands of the original Chord protocol. The main principle, applied by all existing solutions for their hierarchical approaches, is to represent a hierarchical depth level by another tier, which is different than the original tier that lies closer to the base level of those implementations. I will highlight some of the implementations that meet the proprieties previously mentioned:

- The Crescendo [37] solution consists of several interconnected ring implementations, where some nodes from each ring point to each other to obtain an ordered distribution of keys per whole identifier space. Features of load - balancing, fault isolation, hierarchical storage control and storage access, are presented additional to the architecture design.

- The architecture design presented in [38] provides a hierarchical implementation based on two tiers. The base Chord overlay coordinates the second level depth of other overlays. Only the nodes that earned the property of Super Node can coordinate other overlays within the base overlay. One Super Node coordinates the second layer depth overlay through an additional set of finger table and successor list to keep track of the second level depth queries.

- Another approach [39] handles the hierarchy in a concentric manner. The highly reliable P2P system called HIPEER represents the overlay that handles the other hierarchical overlays situated above it.

- The approach used in [36] handles the hierarchy on top of a base overlay. Links are built between several level depths with controlled cost, a lookup operation between two hierarchic overlays being the amount of the total hops needed for travelling to one level depth to another. If a node joins the network, it must first join the base overlay, and then to continue until it reaches the corresponding upper level depth.

Although many implementations pursued the need of handling peers in a restrained framework, separately from the application decisions, there still remains a need for a platform design where several P2P applications can work at the same time interactively or independently. This represents one of the thesis work areas, where an attempt of layering Chord over Chord is pursued, so that a range of applications can work simultaneously on the same P2P overlay network.

3.3. Efficiency model for the P2P Network

The Peer-to-Peer applications available today differ in many aspects such as architectural design, application purpose, network protocols, etc. Although many implementations imposed their own architecture design, there is a need of standardization of such platforms for several applications to interact safely at the same time. Also harnessing the variety of resources within the dynamic environment is one of the priorities that were proven to be a reliable aspect in designing such architectural concepts.

Peer-to-Peer (P2P) applications running over a distributed hash table (DHT) based overlay do not benefit from dominant characteristics of nodes in the network (such as resources and speed). A model proposal that facilitates applications to benefit from the predominant features of a node and also a way through which it can prioritize those features on several hierarchical layers is presented in [41]. This model concept is based on several hierarchical modules identified by an ID, each one being modelled through a set of rules defined by the application at the upper layer. Extensions were made to the original Chord protocol for scaling it under a requested number of interconnected hierarchical modules.

The concept present in [41] enables that every application running on top of a DHT overlay should select between specific properties of certain nodes (bandwidth, shared space, computing power, etc.) within the P2P network. The architecture is developed so that the dominant properties for an application define hierarchical levels in restrained entities over Chord, called hierarchical modules

(HM). Every module is managed through a control file, defined here as a dispatch list, whose lifetime is determined by the valid entries of the nodes participating to that module.

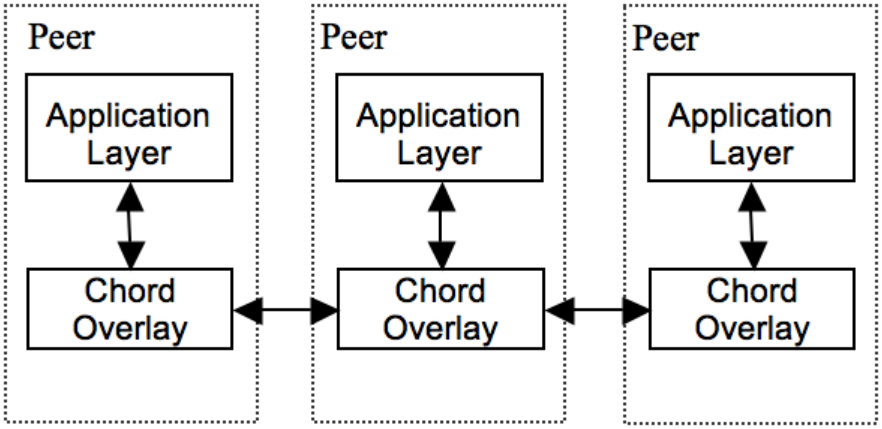


Figure 3.7 Overlay Framework Transparency for the P2P Applications

3.3.1. System Model

The Chord architecture, like many other overlay systems, was designed as a framework for applications situated above the overlay layer, such as file sharing or VoIP applications. An application on top of Chord, handles queries only at key level. The overlay framework underneath the application layer provides transparency between keys and the network transport addresses, handling tasks such as: lookup methods, stabilization, fix_finger_table () procedure, etc.

One hierarchical module from the HM architecture design is managed through the coordination of a dispatch list. A dispatch list is can be identified by the key obtained from hashing the ID of a certain module. A lookup operation between two peers from different modules is associated with the operation of finding first the dispatch list of the queried module, and then following the lookup operation according to the acquired ID and set of rules obtained from that dispatch list.

The hierarchical architecture design serves as an application-driven lookup, because in order to address a query to a certain module, a peer must first find its associated dispatch list, and then it can perform a direct lookup operation to the desired module.

The hierarchical extension of the Chord overlay serves as a framework for the following application types:

- **Location services**: several hierarchical modules can be built according to the validation of Country ID, provided for example from an external service such as MaxMind [41]. Each hierarchical module can host several partitions identified by the Region IDs of the inherited country module.

- **Isolated storage space**: gained through organizing Super Nodes in several hierarchical modules. A Super Node property designates more than average resources, such as: bandwidth, storage space, high uptime, etc.

- **Security:** a certain rule used to create a hierarchical level depth, may serve as a public key for other peers to encrypt data. Only the peers that know the private key can decrypt the available data at this level depth.

- **Task simulations**: for embedded systems over the network. Several tasks cannot complete their activities until other are not finished. One can design a system where dependencies between tasks are ordered from higher priority level to lover ones.

3.3.2. System design

The architecture model [41] is structured according to the hierarchical module validation. There can be several modules implemented, each one of them being identified by a unique ID. Hierarchy depth is set depending on the number of rules that help create such an entity. Each level depth can contain several partition overlays according to the specified rule entry. A rule entry determines a range of values for a certain node property (ex: bandwidth range values, storage space range values, etc.).

Every hierarchical module managed through a dispatch list, which facilitates the operations of joining, failure and lookup. One lookup cost between two different modules is achieved through the number of hops necessary to get the dispatch list of the queried module and the number of hops to get the requested information directly from the resolved node.

The entire multi-ring format runs on top of a traditional Chord overlay. Instead of organizing a range of connections between several hierarchical Chord overlays, the entire multi-ring structure is mapped on top of Chord structure.

The hierarchical architecture design is present in Figure 3.8. Level 0 depth marks the presence of the base Chord overlay network. One single partition overlay can exist at this level and contains the total amount of nodes, through which several hierarchic modules are created. The dispatch lists used to handle operations between several hierarchical modules are stored at this level depth.

Each node is present in the network with a set of rules and a given ID that marks the hierarchical module to which it belongs. A $rule_I$ entry of a certain node will place this node in one of the level I depths, $1 \leq I \leq R$, R = limited. The value of a certain $rule_I$ entry can vary between 1 and M^I value. The $rule_I$ entry value specifies the partition overlay for one level I depth and can be either static (constants) or dynamic (ranges). Hence, a partition overlay designates a smaller number of linked nodes within an overlay.

In Figure 3.9, an example of one hierarchical module built according to the specification of location services provided by MaxMind [42] is shown. Two rules are used for building one hierarchical module such as country and region. The $rule_1$ entry specifies the country name, and partition overlays are formed through grouping nodes with the same country.

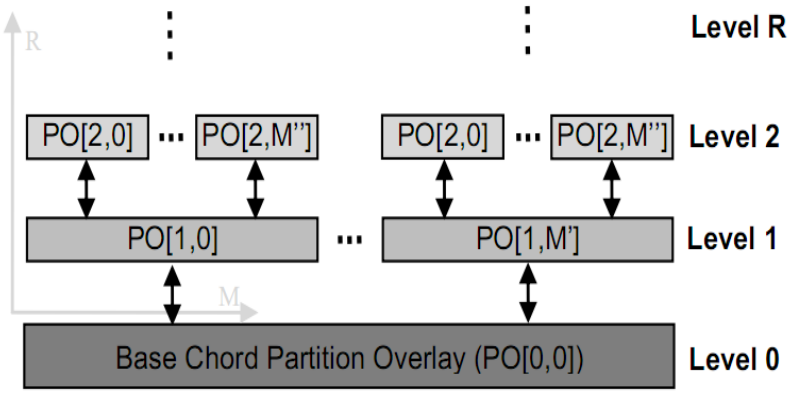


Figure 3.8 Hierarchical Architecture Design

At level depth 2, partition overlays are formed through grouping the nodes from one country according to their region, marking the $rule_2$ entry. The module identifies itself according to one country (the dashed lines), and a single region (the dotted lines).

Continuing with the example from Figure 3.8, level 1 depth has M' partition overlays. Also, level 1 depth marks the beginning of a certain hierarchical module defined in the level 0 depth through its assigned dispatch list. One node can join a certain partition overlay at this level depth, if the $rule_I$ entry value is valid according to this partition overlay specifications and the given ID is part of the hierarchical module created at this partition overlay level depth. Furthermore, the level depth of one hierarchical module is limited according to the R set of rules.

All the nodes needed for creating several hierarchical modules are present at level 0 depth. All the node IDs are mapped under the same ID space, from level 0 depth following level R. Suppose $node_k$ is present at level 0 with ID_k, where $0 \leq k \leq N$, N = total number of nodes in Chord. When $node_k$ joins the upper level overlay according to $rule_1$ value, its hash ID at this point will be ID'_k and $ID_k = ID'_k$. Following this iteration, at level R, $node_k$'s ID will be ID^R_k, where $ID = ID'_k = \ldots = ID^R_k$. Hence, all the ID hashes at level depth 0 will be valid also at level R.

At each level, Chord overlays are formed through linking nodes with the same rule. Notice that one link at a certain level depth, may or may not differ from other links on other layers.

Given the example in Figure 3.9, a two-level depth hierarchical module is presented. The base Chord overlay is formed through linking node IDs from range between 0 and 30 and thus forming level depth 0. At level 1, partition overlay [1,0] is present on top of the base overlay and is marked with dashed connections, according to $rule_1$ entry. This partition overlay marks the beginning of the hierarchical module. Here, node IDs range between $3'$ and $30'$. Although, at this layer, IDs are marked with abbreviation, notice that base Chord ID = ID of level depth 1. At the top level, marked with dotted connections between nodes, partition overlay [2,0] is presented and the number of grouped nodes is smaller than the ones from level 1, level 0 respectively. At this level the link between nodes $12''$ and

30″ appears also in the level depth bellow. This link is not reproduced from level 1 to level 2, this just being mapped from the first level occurrence to the other upper level depths that use the same connection.

Given a set of rules and an ID, one must design a hierarchical module in such manner to avoid overload of keys from a certain level depth. Load balancing is the key factor that allows scaling this architecture to an R level depth. In Figure 3.9 a key that is to be stored with the ID 0, places itself at node with ID 0 in the base Chord overlay, at node 3′ at level 1 and at node 12″ at level 2.

At each level depth, the traditional Chord protocol is used. A node keeps a range of R finger tables and successor lists to route information inside a partition overlay at $level_I$ depth, where $1 \leq I \leq R$, R = limited. The same operations specified by the Chord protocol such as stabilization and fix_finger_table are implemented at every partition overlay where a certain node joins.

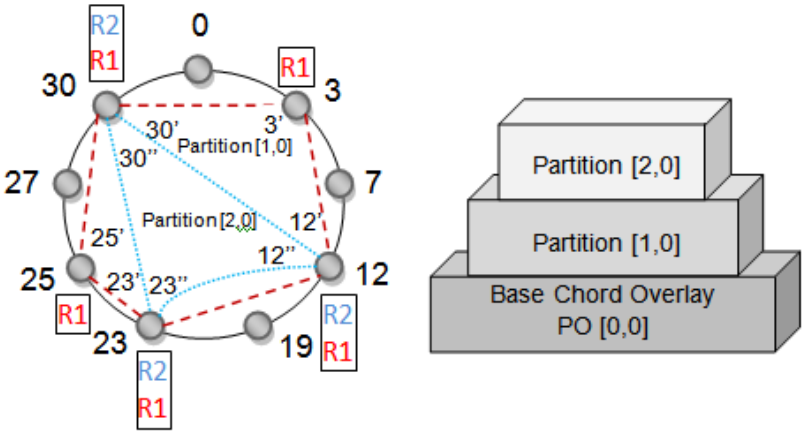


Figure 3.9 Example of a two level depth Hierarchical Module

3.3.3. System Operations

Due to its concentric design, the hierarchical overlay is managed at each level depth through a dispatch list, which is stored at base Chord overlay, level depth 0. The key ID value of a dispatch list is equal to the ID of a certain hierarchical module. Any access that defines ground rules, such as building or reconfiguring hierarchical modules are made from the upper application layer authority.

A dispatch list contains two parts: the first part consists of a set of general rules and the second part consists of entry points for each $rule_I$, $1 \leq I \leq R$. The set of general rules has a global influence over a hierarchical module. In this case, rules are built to control and limit the overlay level depths within a module.

Each node joins a module with a set of R known rules previously set from an external service over the network architecture design such as an application layer. The second part of a dispatch list consists of R other entries identified by the $rule_I$ ID, where $1 \leq I \leq R$. One entry of a $rule_I$ ID consists of a list of M^I values determined by the external service, where M^I represents the range of values from one $rule_I$ entry. The value ranges are used for defining several partition overlays

according to $rule_I$ ID. An entry for the $value_J$ consists of m identification entries, where m represents the size of the finger table from the base Chord overlay; $0 \leq J \leq M^I$, M^I = limited. The identification entry consists of certain information about a node joining the module according to $value_J$ and $rule_I$ ID, such as the ID in the base Chord overlay, a number of successors within the partition overlay and its network transport address. The range of the m identifier varies for each value according to a certain rule is held in an MRU (Most Recently Used) buffer of last joining nodes. The MRU buffer assures the m entries are always valid at a particular moment in time when lookup services are required for a certain hierarchical module.

The second part of the dispatch list is used only for joining, lookup and failure operations. It keeps track of a hierarchical module architecture implementation, and is used for validating the joining nodes according to the known set of rules.

The set of general rules has the same structure as the second part from the dispatch list. Value entries are defined for each rule by the authority of an upper layer application. A node can join an upper level partition overlay only if it matches the according value of the rule (in case the value entry was defined as fixed) or if its value falls inside the range values of $rule_I$.

The hierarchical architecture is designed to act as a framework for other applications. There are two ways of interacting with the network overlay model: static or dynamic. If managed dynamically, an initial set of rules is set, afterwards, the dispatch list completes according to this set. Otherwise, the rules are written manually into the dispatch list at certain points in time. Hence, the properties of the dispatch list are highlighted: limit - through the level depth and control – through the static or dynamic management of the list.

In order to perform lookup operations, a node must have access to all the necessary information. The upper application layer must coordinate a lookup operation between several hierarchical modules according to the ID and the set of rules for a certain hierarchical module. This method of lookup operations confirms that this implementation is more of an application driven network, where the two layers cannot complete operations without the correlation of certain tasks.

A node establishes several connections with other nodes within the module / level depth / partition. To maintain these connections, a node keeps track of a list with as many finger tables as the number of partitions he is being part of. The node can automatically translate itself at a certain level depth to perform queries between partitions.

A lookup operation is performed in two steps. First, the dispatch list is fetched from the base Chord overlay for the queried hierarchical module. Second, the queried module is directly addressed through the information from dispatch list.

Suppose node p from level depth n, partition overlay [n,0], module 1, wants to perform a lookup for a key value at module 2, level depth m, partition overlay [m,0]. Node p uses its finger table according to the base Chord overlay partition and fetches the dispatch list with the key ID 2 (for module 2). It filters the rule identifiers according to the fetched dispatch list and retrieves the number of nodes that matches the requested query. Then it contacts one of the nodes from the MRU buffer and asks for the key for that partition. The contacted node accepts the query and uses its finger table for the requested level depth /partition overlay and returns the successor of the queried key.

When a new node n wants to join the hierarchical overlay, it has to contact the upper application layer to retrieve the rules and module ID according to which it can join the network. It obtains the address of a known node n'_{exist} already joining

the base overlay network and contacts that node. It may ask now for its ID under the base overlay and for its successor according to Chord protocol. It builds its finger table and announces his new place in the base overlay DHT. When all these procedures have finished (also called stabilization procedures), it contacts the node holding the dispatch list according to the ID he received from the upper application layer. Node n fetches the dispatch list and retrieves the m nodes marked as a perfect match to its current state of rules and continues its joining operation to the upper level depth pointed by its set of rules. If one entry of a certain rule doesn't fit the requested value, this will force node n to remain at the current level depth until that specific entry matches the rule. After the joining operation has been performed, node n updates the dispatch list according to the level depths / partition overlays where it joined. The update consists of several identifier entries according to the values of the set of rules where the match was found. An identifier entry includes the node ID in the base Chord overlay, the number of successors from a certain partition overlay and the network transport address.

The joining procedure will proceed from the lowest level depth to the highest. When node n joins a partition at a certain level depth, it only updates the finger table from this partition overlay and informs the other nodes of its presence. If the link, established at a lower level depth, is valid for the next upper level depths, then this link is automatically mapped at the level where the common rule ends. Each partition overlay is responsible for load balancing its data. This means the new node will be responsible for a list of keys, each entry in that list corresponding to a certain level depth partition overlay in the hierarchical module.

When failures occur over the overlay network, the base Chord overlay is the first to assure the integrity of hierarchical modules. When a node n leaves the network, it also leaves the hierarchical partitions overlays it is being part of. When a node at a certain level depth calls the stabilization procedure and confirms that node n is not part of the network, it checks the status of the dispatch list that handles the current hierarchical module. If node n, which isn't present in the network overlay anymore, appears in any of the m entries of the R rules of the dispatch list, it will be replaced by the node who first noticed node n disappearance from the network. In this manner the integrity of the dispatch list is maintained.

The worst case scenario occurs when levels depth in hierarchical modules cannot be established anymore. In this case, the keys are translated to the base Chord overlay and marked so that if a level depth can be established again, the nodes on that level must retrieve all the information according to the new level depth partitions overlays that are newly formed. If one level depth fails the keys are moved to the below level.

If the base Chord overlay is fragmented, then the architecture maintains the connections at upper level depths. Suppose that a hierarchical module is structured according to the MaxMind [42] with the following rules: R1 = Country, R2 = Region. This example scales to a two level depth hierarchical module. Suppose that on level depth 2, two region partitions are present, and for some reason are held by different Internet service providers (ISP's). If any of them goes offline, the failed partition is still available at level depth 2 through the connections within its Internet service provider (assuming that a certain local area network is still up). At level depth 1 and base Chord overlay, the connection links are marked with the offline status. When the connection is re-established between the fragmented partitions and the base Chord overlay, the nodes will remap themselves at the level depth 0 and 1, respectively.

The dispatch list is handled at the upper application layer for replication purposes. The lifetime of a hierarchical module is valid until there are no more valid entries in the dispatch list.

3.3.4. Performance Analysis

This section provides an analysis of the lookup and maintenance costs of the architecture design present in [41] and its comparison with Chord. The following assumptions serve as a starting point for our cost analyses [19]:

- The total number of nodes in the base Chord overlay is N.
- The total number of hierarchical modules mapped over the base Chord overlay is M'
- The total number of partition overlays (PO) at a certain level depth is $PO = \frac{M^{I}}{M^{I+1}}, 1 \le I \le R$.
- The total number of nodes in one partition overlay (NPO) at a certain level depth is:

$$NPO = \frac{N}{\prod_{I=1}^{R} M^{I}}.$$

- The probability that a node possesses a copy of a dispatch list for lookup operation purposes, with a valid entry for a certain level depth is

$\mu_{I}, 1 \le I \le R$.

When a node performs a lookup operation to a certain hierarchical module, it first acquires the dispatch list of that module. According to the fetched dispatch list, it performs a direct lookup operation to the desired level depth of the requested hierarchical module. By using the standard lookup operation, the node acquires the dispatch list in $\frac{1}{2}\log N$ hops, according to the Chord protocol specification [19].

The total amount of hops that a node requires to perform a lookup operation to a desired level depth of a certain hierarchical module (HM) is:

$$Lookup_{HM} = \frac{1}{2}\log N + \frac{1}{2}\log \frac{N}{\prod_{I=1}^{R} M^{I}} \qquad (3.1)$$

The derivation of Equation 3.1 is a consequence of the property of a node to translate its link connections at a certain level depth. Hence, the routing cost between several level depths within the same hierarchical module is expressed in terms of translation between level depths instead of going through several hops to get to the desired level depth. This also shows that the number of hops of any lookup operation is proportional to the number of hierarchical modules mapped over the base Chord overlay.
If a node has performed lookup operations to the same hierarchical module and already has a copy of the designated dispatch list, then Equation (3.1) depends on μ_I:

$$Lookup_{HM} = (1 - \mu_i)\frac{1}{2}\log N + \mu_i \frac{1}{2}\log \frac{N}{\prod_{i=1}^{R} M^i} \qquad (3.2)$$

The maintenance cost of the Chord overlay is determined by the periodic call of *fix_fingers()* and *stabilization()* procedures [19]. For each partition overlay within any level depth, our hierarchical module implementation conserves the maintenance costs as present in Chord. Because the proposed architecture design does not aim to build hierarchical modules at a 1:1 ratio, overall maintenance costs tend to a minimum value at higher level depths.

3.3.5. Simulation and Experimental Results

The hierarchical architecture design was simulated through using the Oversim simulation environment framework over the OMNET++ IDE [43], which is designed according to the application and overlay network layer. The application layer is built over the overlay network and provides transparency between keys and network transport addresses.

The dispatch list is implemented at the application layer, and coordination with the overlay takes place through remote procedure calls. Network parameters such as hierarchical module count, rule count, rule entries, churn rate, are selected at runtime. The probability of a valid entry in the dispatch list is simulated through the churn rate of the simulation framework and through a parameter given at runtime. A pre-generated file was used for naming hierarchical modules within the Chord overlay. In order to compare the hierarchical architecture design with Chord, every lookup operation was set first time for the hierarchical module and a second time for the base Chord overlay according to the same queried key.

Simulation results were collected from a number of 100, 200, 500, 1000 and 2000 nodes. Only three level depth hierarchical modules with two different design ratios were used.

Figure 3.10 presents a hierarchical architecture design implemented according to the ratio of 10:5:2. Ten hierarchical modules are created according to the level depth 1, each one holding 5 partition overlays at level depth 2, and at level depth 3, 2 partitions are present. The probability of a node holding a valid copy of a dispatch list is set to 0.5 and 0.3 respectively. The mean hop count of our architecture design starts to be higher than Chord beginning with the number of 2000 simulated nodes. The mean hop count for the level depth 1 exceeds Chord beginning with the count of 500 nodes. This can be explained through the number of

hops that a node has to take to reach the queried key. Both level depths 2 and 3 show better performance than Chord in our architecture design.

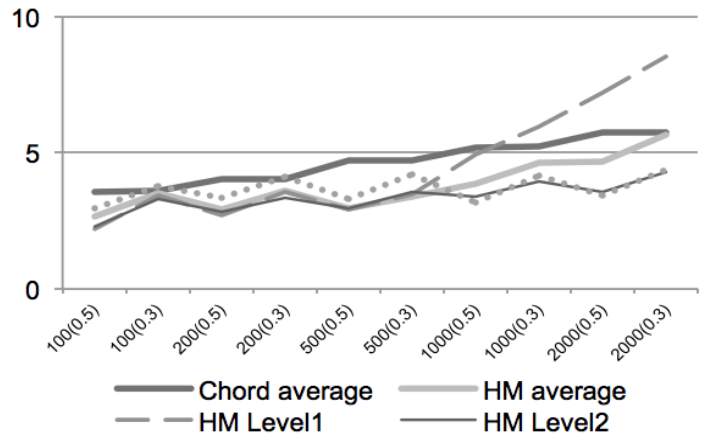


Figure 3.10 Hierarchical Architecture Design for 10:5:2 ratio

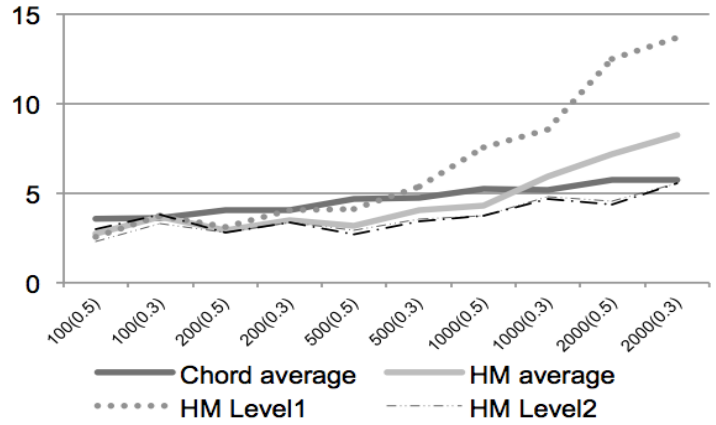


Figure 3.11 Hierarchical Architecture Design for 20:10:15 ratio

In Figure 3.11 hierarchical modules are created according to the 20:10:5 ratio. The same probability is used here for one node holding a valid copy of dispatch list for performing lookup queries. Because the number of hierarchical modules is greater than the previous simulation, it performs better until the number of 2000 nodes is reached. The mean hop count remains below the Chord, and matches it at 2000 nodes. The mean hop count for the level depth 1 exceeds Chord beginning with the count of 1000 nodes in this case. The architecture design shows good performance if the number of nodes, within hierarchical modules, is equally proportioned throughout the architecture levels.

3.4. Peer Availability – Uptime Case Study

One of the many issues that Peer-to-Peer network architectures are confronting with is represented by handling the availability of joining peers. Although early implementations of P2P concepts were relying entirely on simulations to adjust their internal protocols, new approaches base their implementations on the case study of measurements obtained from previously developed systems.
The host availability is determined by the uptime property of every node that joins a P2P network system. In an unpredictable environment, where every peer can join or leave the network at any time, it is a great challenge to manage the information safely. This generates the need of caching and replicating data content across such platforms.
According to [44], host availability can be defined as:

"The degree to which a system, subsystem, or equipment is operable and in a committable state at the start of a mission, when the mission is called for at an unknown, i.e., a random, time".

The host availability cannot be always determined by the intention of the user to join or leave the network. There are factors that contribute to the increased/decreased uptime level of a joining peer, such as: exchanging information in a trusted manner, security issues, privacy and poor incentive mechanisms. Further, my expectation of peer availability changes drastically according to the P2P application type: a file sharing system has lower uptime expectation from its peers than a VOIP application across the same distributed system.

The work available in [45] provides an accurate case study of the Kazaa network session. In Figure 3.7 (Left) measurements of handling connections between ordinary and super nodes were provided (ON-SN and SN-SN). The range of connections varies for both cases: 100 to 160 for ON-SN session and 30 to 50 for SN-SN session. Although there is a slight tendency to a constant, the individual connections change frequently. The average duration of one individual connection is 34.4 minutes for ON-SN and 11 minutes for SN-SN.

The neighbour selection [45] of Kazaa entities (SN or ON) is made according to one of the following:

- Low workload of the super node involved in one of the ON-SN or SN-SN connections.

- Closer location based connections in terms of:

  - Round-trip time (RTT) measurements. A connection can be established within a value range of RTT less than 50 milliseconds.

  - IP subnet mask matching. A connection between any of the ON or SN can be established if their IP prefixes match.

Trough the provided analysis in [45] I conclude that any newly joined node to the Kazaa application will establish several connections in any of the SN-SN or ON-SN directions, until an adjustment of geographical area distribution takes place.

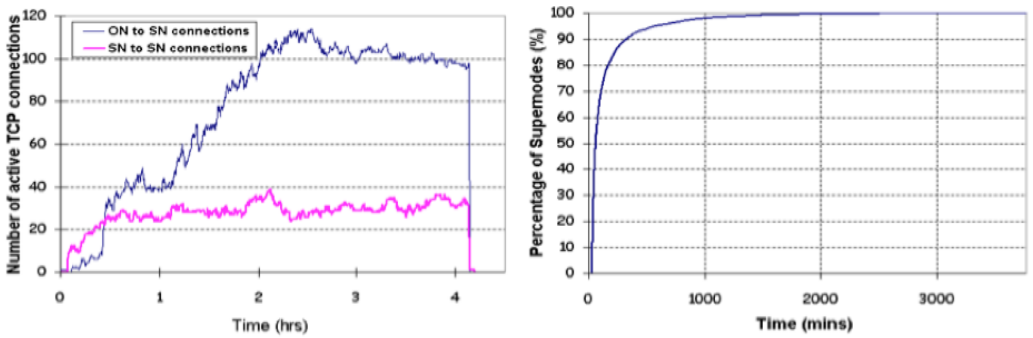


Figure 3.12 Kazaa Session Evolution [45]

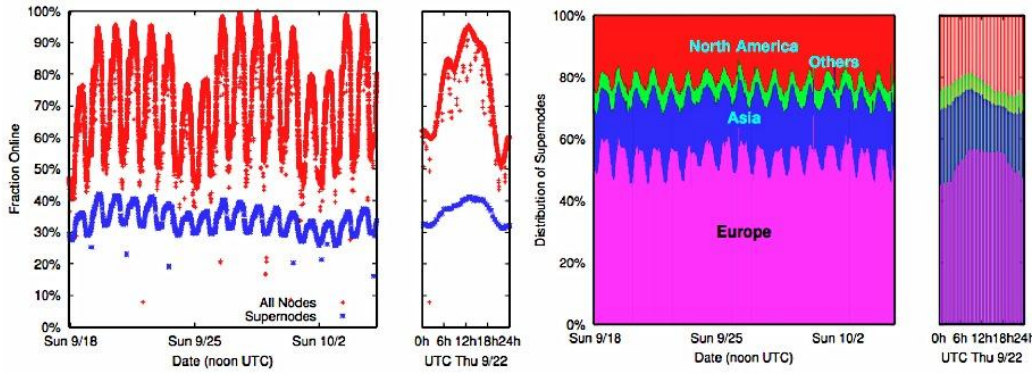


Figure 3.13 Skype Session Evolution [46]

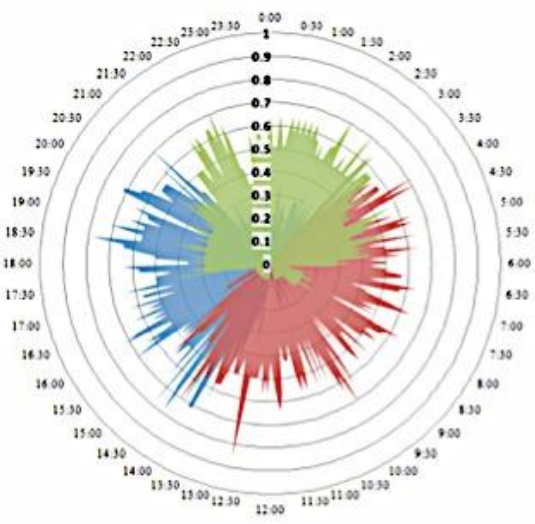


Figure 3.14 BitTorent Session Evolution [47]

After this limitation is reached, connections can still vary but only within the same geographical distribution range. This approach proves that Kazaa carefully places peer operations within a range geographical distribution limit to prevent network overload.

In Figure 3.7 (Right) the super node uptime was analysed. A number of 965 unique super nodes were monitored over a period of 65 hours. As a conclusion, the average SN uptime in the Kazaa overlay is 149 minutes (2.5 hours).

The work presented in [46] focuses only on Skype super node churn analysis. Although Skype architecture design relates to Kazaa [8], peer activity differs significantly in terms of uptime and connection stability. Figure 3.8 (Left) shows that the number of super nodes is more stable than other Skype clients (ordinary nodes). A range of 30 to 40 connections can be established between SNs, with an average value of 35. Further, super nodes prove to have a diurnal behaviour, causing a reduced usage of 40-50% at night. The average super node uptime rises to 5.5 hours, higher than most of the available P2P sharing systems.

Figure 3.8 (Right) shows the geographical distribution of super nodes. The peaks present in this figure confirm the diurnal behaviour of super nodes. Although nodes sign out mostly at night, there is a significant difference in the overall SN availability. This geographical distribution confirms that some regions adopt more or less the facilities of P2P architectures.

The work presented in [47] analyses the BitTorrent [48] client availability. BitTorrent represents one of the most popular P2P sharing application available today. Measurements were made over a number of 10,000 nodes across 191 countries. The distribution of nodes was more pronounced in three countries (US, UK and Canada) than others. From 10,000 nodes that were observed in 2 weeks, an average availability of 28.39% was obtained, with a median of 15.67%. Figure 3.9 shows the variation of availability of 3 selected peers within the output average of 28% available peers. The circle represents a wall clock (24 h) and the length of the radius refers to availability. The experimental process shows that the red peer is available between 03:00 – 15:00, the green peer from 18:00 – 05:00 and the blue peer from 14:00 – 21:00. As an overall output result, Bittorrent clients have an average availability of 2.8 hours.

In table 3.2 a summary of the previously analysed Peer-to-Peer session characteristics are presented:

Table 3.2 General Peer-to-Peer Session Characteristics

	Average Uptime	Geographical Distribution	Explore Peer Heterogeneity	Established Connections
Kazaa	**2.5 h**	**Yes**	**Yes**	**20-50**
Skype	**5.5 h**	**Yes**	**Yes**	**30-40**
BitTorrent	**2.8 h**	**No**	**No**	**0**

Although Kazaa and BitTorrent are two sharing applications that differ in terms of architectural implementation, both have a similar average node uptime. The main architectural difference between the two applications is represented through the way that peers are sharing their resources. BitTorent determines peers to create a *torrent* file pointing to the address of the host that wants to share any information. The torrent file is then uploaded on a web server that facilitates the operation of making the file available to other clients that query that information. BitTorrent uses the third tier (web server) to centralize a small amount of

information that can lead to the files available for download on the first tier (Peer-to-Peer layer). Kazaa determines peers to participate more actively in the sharing process by centralizing the information needed for lookup operations on nodes with above average computing resources. The environment held by the super nodes is not comparable with the server's property of guaranteeing data availability and consistency. Also the limit imposed by the geographical distribution, decreases the availability performance on the Kazaa P2P platform.

Skype proves to be a reliable P2P solution trough the following:

- **Security**. Skype secures both voice and message operations across the P2P network.
- **Information Exchange**. Occurs in a trusted manner, by precisely addressing the user to whom information is sent or communicated.
- **Incentive mechanism**. Determined by the need of communication. Communication represents one of the key elements that is at the base of development of society.
- **Privacy**. Skype collects data only to ensure better quality of the VOIP service and better user experience.

Skype represents one of the Peer-to-Peer applications that perform well in terms of performance and host availability. Although its architecture design is implemented across three tiers, the login server is used just for storing and authenticating user IDs. The entire mechanism of VOIP and messaging is supported by the P2P structure: ordinary nodes and super nodes.

Through the provided analysis of host availability within P2P networks, I conclude that a reason in obtaining a high uptime in a dynamic environment is represented by the natural desire of communication and information sharing. The trust gained by the environment generated by the P2P mechanism also plays a major role in attracting new peers to the network.

3.4.1. Replica Placement Algorithm using Uptime Prediction Methods

The Peer-to-Peer architecture concept denotes an unstable environment sustained only from participants that can leave or join the network at any time. To provide a stable holder for the data content that is to be stored across such environment type, many algorithms were developed to provide, with a certain level of accuracy, the moment in time when a peer is available in the network. The first attempts in handling data consistency and availability across P2P networks were relying on replicating information across a random number of peers. Thus, this method offered a quick answer to the issued problems, the P2P system architecture suffered from overall performance loss: high bandwidth resource usage, increased computing power generated through the constant need of handling additional replication techniques across the network, etc.

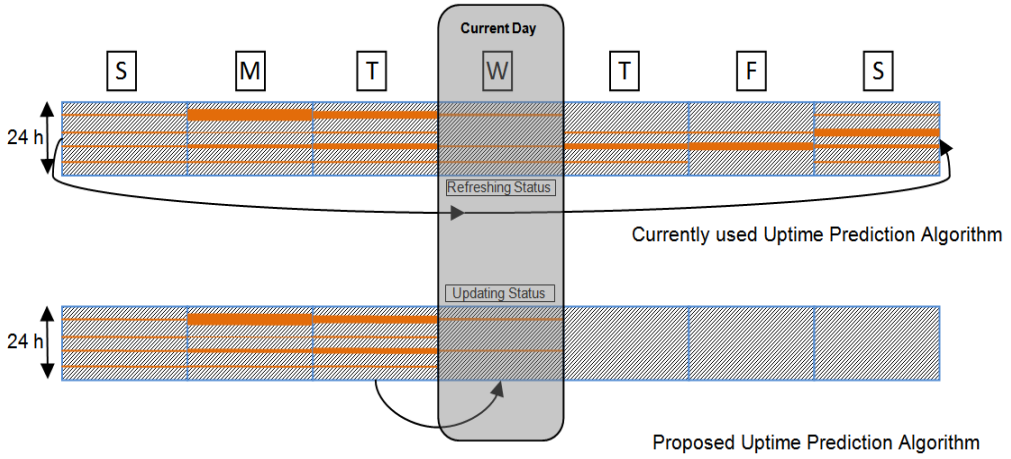


Figure 3.15 Uptime Prediction Algorithms

The replica placement algorithms were developed to minimize the effort needed to handle both data consistency and availability across P2P networks. Through precisely aiming at peers with an increased time spend across the P2P network, the replica algorithms were able to minimize the number of nodes involved in the caching process and also to increase the level of both consistency and availability of data. One of the algorithms that perform in such manner is described in [47], providing a good support for applications to safely store data across unstable network environment.

The authors from [47] provided a probabilistic model of handling replicas across structured P2P networks by building a peer availability table (PAT). The algorithm breaks down the time interval of one week into several slots of 5 minutes length. Through this measurement slot a day is divided into 288 units and a week into 2,016 units. PAT indicates the peer availability at slot level, every entry being computed according to the formula [47]:

$$AV_I = \frac{OnlineCounts}{VerificationCounts} \quad (3.3)$$

Through Equation 3.3, the provided algorithm can precisely specify the user's joining habits to the Peer-to-Peer network during a time interval of one week. Analysing the user habit of joining the network, the authors refer to the precise moment when the user usually spends most of its time to the P2P network. The Online Counts refer to the current registered online status that one peer personally keeps track and the Verification Counts are handled from other peer neighbours from the network. The whole algorithm performs as a PING mechanism, where merely the PAT table is adjusted from a restrained number of peers for a better accuracy of the algorithm. The Verification Counts fields are refreshed every 5 minutes from both the subject and neighbour sides.

The presented algorithm provides a good foundation of replicating the data among an interval of one week providing two replication techniques: across peers with PAT similarity (consolidating data availability) and across peers with different PAT configuration for a better data availability.

Although the smallest time unit has a five minutes length, the overall observation window is handled into an interval of one week. This algorithm handles well replication techniques across precisely delimited time spaces, but the observation window needs to be adjusted for more accurate and prompt replications decisions.

In a Peer-to-Peer network application that needs to handle critical data promptly and safely on such unstable environments, the overall observation window must be restricted to the smallest unit possible (Figure 3.15). The mailing system generates a communication platform where the priority is first set on finding and notifying the recipient for the already cached email content. If the replication algorithm is highly accurate on a time window that requires an increased number of nodes for caching the email content, several issues can occur such as: inaccurate usage of the available time slots due to the limited number of nodes available of caching, high bandwidth usage due to the multiple replicas involved in the storing process and increased delay in notifying the recipient for the new incoming email.

The changes that are specific for a mailing architecture across the P2P environment require decreasing the time window from one week to few days of observation and also decreasing the time slot interval for reaching this performance. An average mean of the computed availability per slot daily increases the prediction performance of such algorithm.

Figure 3.15 presents the advantages brought by the new proposal of uptime prediction algorithm. A time interval of seven days is analysed through implementing both the uptime prediction algorithms present in [47] and [61]. The grey indicator highlights the current day according to which the uptime is refreshed or computed accordingly.

Although the algorithm used in [47] requires a time window of one week for refreshing or updating the uptime status, it requires double the amount of time for refreshing the status of one day. Hence, the uptime prediction computed for a certain day may not have the same prediction a week later. When a file replication takes place for a period of one week, faulty time slots can be used for this matter. Through this implementation, the replicated file may be unavailable in a time interval established from the faulty time slots computed or chosen for this matter.

The proposed algorithm is similar to the one presented in [47]. It uses a prediction window of one day. The algorithm uses the criteria of updating the time slots daily by using the information previously computed a day before. To provide the best prediction results, the algorithm requires a time span of 5 days. The algorithm is best fitted for handling the replication process according to a server-centric implementation of P2P overlay design.

The new proposal computes the uptime prediction according to several time slots, each one having a span of 1 minute. A weighted average of previously computed uptime prediction is computed every day passes by.

Figure 3.15 shows that the new proposal takes in consideration only the time slots that show a constant value in time. Any variation of the time spent over the P2P overlay network cannot be taken into consideration for a good prediction status of this algorithm. The new proposal can be implemented on applications that require a prompt and quick replication process across the Peer-to-Peer overlay network.

By using only the time slots that do not excessively vary in time, the algorithm provides a precise uptime prediction suitable for file replication operation across Peer-to-Peer overlay networks.

4. Current Mailing Architectures

Initially designed for academic purposes, the electronic mail (*email or e-mail*) application has become one of the most used tools that modern society has adopted at daily basis. The traditional mailing mechanism relies on the server-centric structure where the mailing client (*mail user agent* - MUA) connects to a server to perform email operations such as sending, retrieving or querying information. The term server describes a complex system in handling email operations across several clusters in a distributed manner. The mechanism of handling distributed mailing tasks is described by the *mail transfer agent* (MTA), which assures the attributes of performance and quality of service in terms of data replication, location services, network availability and load balance.

The traditional mailing system has reached a mature stage through constantly improving several mechanisms such as the protocol standard format [49], network topologies used in handling distributed tasks at cluster level, computing system requirements and storage facilities. Although improvements have been made, there are also scenarios that overcome the current mailing system model: accessibility issues when the gateway lies behind a link that has been severed or flooded, storage issues and processing stress due to multiple mail attachments [10][11]. The constant need for scaling mailing architectures according to the large number of service requirements, implies also higher costs in terms of dedicated buildings and specialized trained personnel for managing, network bandwidth latency and topologies for handling distributed tasks, computing resources, etc.

Peer-to-peer mailing architectures were developed in response to the high costs and numerous issues of handling client-server mailing infrastructure. Through this implementation design, every participant to the mailing system has to share some of its computing resources, such as bandwidth, computing power, storage space, etc. But this implementation also has some structural flaws: due to peer member behaviour (uptime is unpredictable), it is very difficult to handle a complex architecture design like the mailing system, which implies storage space, data availability, bandwidth, and computing power.

Recent P2P mailing solutions were developed across structured and unstructured platform implementations. The solutions implemented across unstructured platforms performed better on high churn rate (low uptime expectations of certain peer) because the adopted model was able to carefully select peers with above average resources across network for sustaining the whole application backbone (mailing system services). The structured platform was developed strictly as a framework for these P2P applications. Basically this framework provides an application programming interface (API) for the applications running virtually at the upper layer. This solution has solved the issues raised by the unstructured model in terms of limitation of queries for a better bandwidth latency usage and any peer could be directly addressed within the identifier space. The mailing architectures developed on this platform were more complex and efficient implemented than the hybrid models in terms of mailing operations, bandwidth usage, architecture design and implementation.

Considering this brief introduction, a thorough analysis of current mailing architectures will be provided in this chapter. A focus is mainly set on the server

centric architectures (traditional mailing systems) and Peer-to-Peer alternatives to the ones exiting today.

4.1. Traditional Mailing Architectures

Traditional mailing architectures rely on a server centric design, where the whole mechanism of sending and retrieving email content can be divided into two main processes: the mechanism of handling client operations (editing, formatting, transmitting email content) and the mechanism of handling inter-process communication by the server side. Although mailing architectures rely on simple architecture basis, a set of complete tasks is required for handling inter-process communication between random parties. The protocols used in handling intercommunications of such processes were categorized as standards from the Internet Engineering Task Force (IETF) [50].

The tasks implemented across mail servers for handling email communication [51] are shown in Figure 4.1:

- **Mail User Agent (MUA)**. Represents the mail client application that enables a direct interaction with the user. Facilities of editing, receiving and sending email content are provided by such applications.

- **Mail Retrieval Agent (MRA)**. Is closely related to MUA and most of the modern systems are handling it within the MUA process. It is used for retrieving email content from the main mail store by one of the available POP or IMAP protocols.

- **Mail Transport Agent (MTA)**. Provides message delivery from third parties such as MUA or other MTAs. Email messages typically travel several MTAs until the destination is reached.

- **Mail Submission Agent (MSA)**. Specialized form of MTA. It accepts mail submissions from MUA side and handles any specialized processes that may be required.

- **Mail Delivery Agent (MDA)**. When the mail message reaches destination, the mail delivery agent specifies the inbox location holder for storing the content.

Figure 4.2 provides a thorough analysis of how mail transfer is implemented across the Network. When the user finishes editing his email, the MUA client is used for sending the content through contacting the nearest assigned server for that purpose. Considering that the receiver's destination differs from the sender's in terms of mail service provider, the assigned server (MTA) performs a mx-lookup to retrieve the mx-records from the Domain Name System DNS [52] service. A domain system can point to several destinations, having several mx-records to the queried destination mail service. The MTA connects to the server, from the acquired mx-records, with the highest priority and delivers the mail content through the SMTP protocol. If multiple mx-entries have the same priority, the MTA chooses the server randomly and establishes the connection. When the mail content reaches the

destination MTA, the email content is placed through the MDA to the mail message store. The retrieving operation of email content is performed from the MUA client application through the MRA according to POP or IMAP protocol.

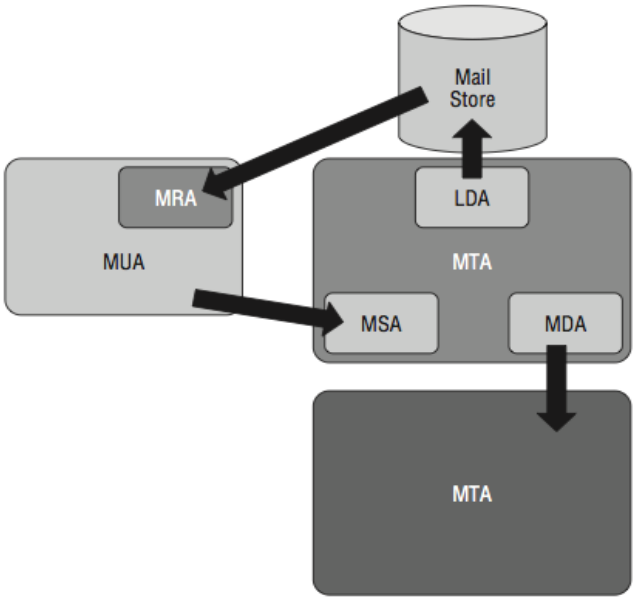


Figure 4.1 Components of an Email System [51]

When performing the mx-lookup operation, a Mail Relay address can be received from DNS service. The Mail Relay is often used to set an additional MTA for performing filter operation for spam or virus scan. After the filter process, another MTA is reached through performing the same operations of mx-lookups for finding the recipient destination.

The paper present in [53] provides a mailing system classification according to the design decisions that affect or help the users to access their email account:

- **Store and Forward Servers**
- **Server-Only Mail Repositories**
- **Client-Side Caching Systems**

Most of the Internet mail service providers are represented by store and forward mail systems. Through this approach the email content is kept on the server side until the process of retrieving occurs from the MUA side. The protocol used in this operation is Post Office Protocol. This solution presents two advantages: very little processing is involved in handling or caching email content over a limited period of time and through caching all the email content locally, the users have access to their mail even if no connection to the Internet is available.

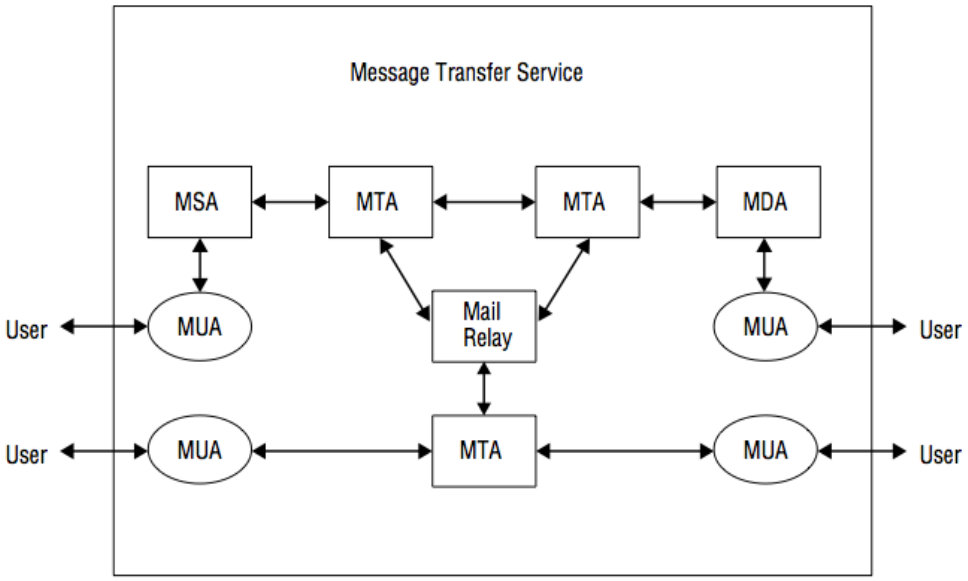


Figure 4.2 Mail Transfer Service [50]

Unfortunately this approach of handling email content, partially on the mail server and permanently on the personal computing resources, has also its downsides. If the mail service is reached from different personal computing resources for the same user account, the managing and administrating mail inconsistency occurs on the user side. Further, replication and restoring email content occurs also on the user side, this process requires heavy computational resources from the personal computing resource. Another issue arises when no connection can be established to the mail server and users cannot have access to their inbox holder for retrieving new incoming mails.

An alternative to store and forward approach is to store all the messages on the server side. This solution is currently available on web based mail servers such as Yahoo and Hotmail, through the traditional enterprise mail servers such as Microsoft Exchange and Lotus.

The advantages of Server-Only Mail Repositories solution are the following:

- A consistent view over the mail repository is achieved due to accessing the mail from the server side.
- Data restoring and replication occurs on the server side.
- Several features are available though the mailing server side computing resources. Features such as handling email content from mobile devices are available on Server-Only Mail Repositories.
- Notifications are available through mail push services.

The drawbacks of the Server-Only mail system are as follows:

- The system is heavily dependent of the network. Failures over the network will prevent users to connect to mail servers for accessing their mails.
- The server-only architecture has a limited number of network gateways. A failover occurs when a large number of connections are established to the server. This mainly results with a slow network connection for sending/retrieving email content.
- Scalability to handle a large number of users is added over the expenses of feature richness.

The Client-Side Caching solution combines both the features of previously described systems for a better user experience. The assigned protocol in handling email content over the network is Internet Message Access Protocol (IMAP).

The advantages of this approach includes the ones of previously described mail systems such as disconnected access to mail messages, consistent message store view from different clients and search facilities.

The downsides of Client-Side Caching Systems are as follows: increased development time and bugs facility due to the client-server synchronization protocol, runtime performance is reduced due to the overhead of synchronization operations and high storage facilities decreases the number of features over the mailing service.

4.2. Distributed Mailing Architectures

Attempts of avoiding the drawbacks of server-centric mailing architectures were made in both the software and hardware directions. Software solutions were able to harness computing resources over the Internet through the Peer-to-Peer concept and hardware attempts were focused in handling resources in a distributed manner at cluster level. Regardless of the distributed approach in developing mailing architectures, the process is similar for every implementation: adapt the traditional processes (agents) to the newest implementation. The agents of retrieving, delivering, transporting and submission are redefined according to the requirements or features of the distributed approach.

The software attempts of developing mailing architectures across P2P networks were focused first on generating the stable environment for performing mail operations such as sending, receiving, storing and replicating email content. The second requirement was to achieve performance on the distributed network platform through promoting nodes with above average resources across the network and by harnessing the benefits of both the structured infrastructures and systems. A classification of such implementations can be made according to the P2P network approach:

- **Unstructured**. The mailing architecture design in [54] was developed over a hybrid P2P network design. The proposal offers authentication and location services under the coordination of a server-centric entity. The network

architecture is structured according to community validation, each community consisting of a number of nodes linked to a super node. The MTA property is fulfilled from the super node side, all messages travelling first at this layer and after that being forwarded to the other nodes linked to the MTA node.

- **Structured Infrastructures**. Distributed mail architectures were developed above the framework provided by such infrastructures:
  - The approach used in [55] represents one of the best solutions concerning the P2P mailing architectures. The proposal was developed over the Chord overlay [19] placing inboxes at a precise key in the identifier space. For security issues, the authors employed the services of an external certificate authority, each user being able to identify itself for retrieving mail data over the P2P network architecture.
  - The solution presented in [56] was developed over an overlay network layer. It offers a pull-based solution, where each peer keeps track of the mail content marked for sending purposes and places over the overlay only a notification for the receiver to download the mail content from the sender side.
- *Structured Systems.* The solution presented in [57] was developed under the mobile-object paradigm. The mailbox is represented through an object that travels on the live network to ensure data availability. A second mobile object defined here is the dispatch unit, which holds information about the available active machines on the network. A computer system that goes offline must first upload the mailbox objects to the available systems on the network specified by the dispatch unit.

The hardware implementations of distributed mailing architectures were relying on dedicated hardware equipments across which software mail operations were developed. The solution found in [53] is developed on such preliminaries, having an implementation where scalability is highly proportional with the costs involved in managing the structure.

4.2.1. NinjaMail Architecture Overview

The NinjaMail architecture design was developed on top of the Berkeley's Ninja cluster architecture (Figure 4.3). Its design is built to provide users with highly available, scalable and feature-rich services through a wide variety of access methods.

Data storage and data replication is handled over Distributed Data Structures (DDS), which administer cluster-wide data replication and synchronization for metadata on Ninja infrastructure and other applications built on top of it.

Ninja is implemented in java language, designed in an object oriented manner, where each node is running a JVM, housing the management unit (vSpace)

and other distributed applications. The distributed design yields from the methods applied at communication level, between "worker" objects. Each serialized object has a finite state life time, governing on a single thread allocated by the Ninja architecture.

At cluster layer design, NinjaMail presents several API's for access and extension modules. At Ninja's core module, storing operations are implemented such as: saving and retrieving messages, updating message metadata and performing simple per user message metadata. Access modules are used for communication between users and intercommunication between Ninja clusters.

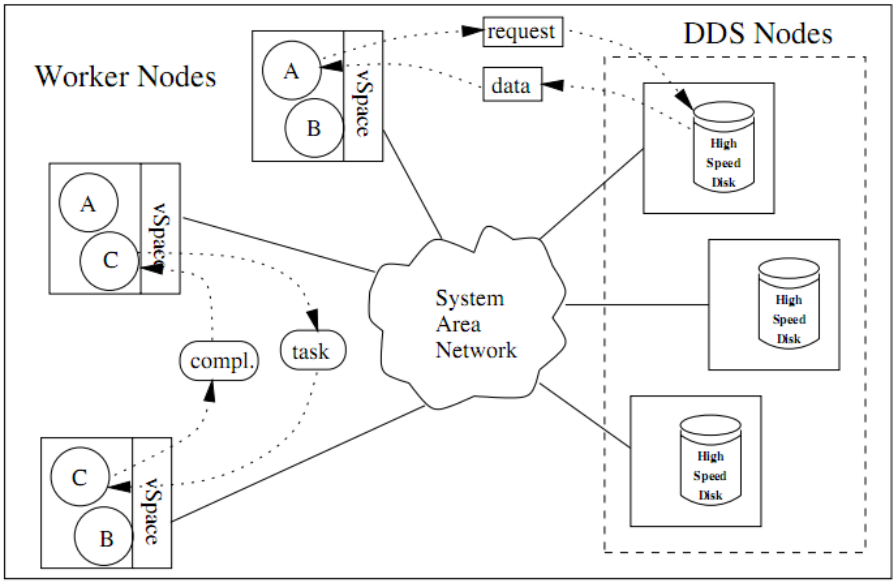


Figure 4.3 Ninja Architecture Overview [55]

NinjaMail is a response to handling the scaling of email use across the Internet, providing reliable, high performance and feature – rich services to users. Unfortunately this solution is expensive and hard to maintain, because dedicated hardware and dedicated maintenance is needed to keep this system operational.

4.2.2. Decentralized Electronic Mail Architecture Overview

The Decentralized Electronic Mail (DEM) is built on a distributed application middleware (Oceanstore [27]), being not hardware platform dependent and maintenance is fulfilled from the mailing service itself. The software implementation makes use of object serialized structures implemented in FarGo [58], which offers several services:

- **Explicit migration**. The application can explicitly request to relocate data structures over other specific hosts, while conserving correct state and execution.

- **Implicit migration**. Migration of components will execute in a developer – define collocation relationship manner.

- **Location transparency**. The application manages it components through valid pointer references, not to real location information.

- **Monitoring**. Object location and migration can be monitored through the application's API.

- **Reference construction**. An enhanced reference can be constructed in such a manner that remote objects can be found on the fly.

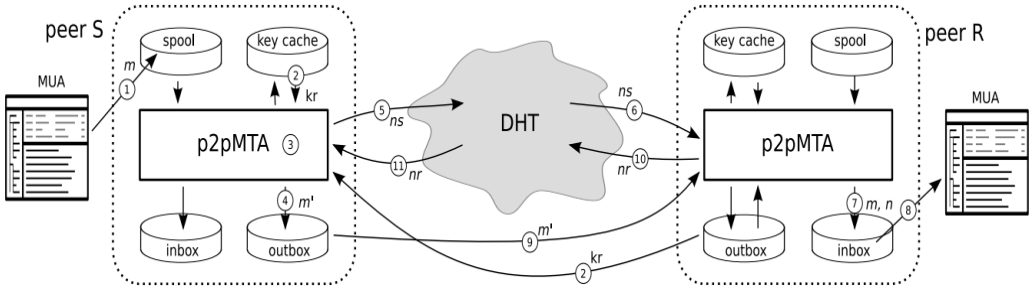
All the primary components of the DEM are mobile objects. The mailbox itself is a mobile object that travels on the live network to ensure data availability. A second mobile object defined here is the dispatch unit, which holds information about the available machines active on the network. A computer system that goes offline, must first upload the mailbox objects to the available systems on the network specified by the dispatch unit.

This solution is one of the best solutions presented, because it runs independently from any hardware architecture. Disadvantages are present here also, because the mail service keeps serialized all the mail information alive. Travelling to one node to another, message information makes use of large amount of bandwidth and computation power. The system doesn't predict which elements are offline and which aren't.

4.2.3. A Pull Based Peer-to-Peer Mailing Architecture - Overview

The mailing system presented in [56] is developed across the Chord overlay. This approach handles email content locally, signalling the receivers through notifications stored across the DHT based overlay. The solution is similar to the file-sharing concept by handling email content through a "pull based" approach. The email content transfer occurs directly between the sender and receiver without a central authentication or storage servers.

Each peer has both connectors for SMTP/IMAP protocols and act as a daemon for local mail clients (MUA). The P2P mailing system has a running client on each computing resource that facilitates connectors for other traditional mail clients such as Outlook, etc.



4.1 Sending/Receiving an Email [56]

Considering the traditional internal processes, each peer handles the operations of a mail transfer agent (MTA) through implementing the following (Figure 4.4):

- ***Spool Area***. Contains the sent mail messages from the MUA side.
- ***Inbox Area***. Contains the inbox holder of newly arrived mails.
- ***Outbox Area***. Holding email that are waiting to be pulled directly from the receiver.
- ***Key Cache Area***. To store public keys for encrypting the email content. The public keys are accessible to other peers within the overlay while the private key is safely stored on the local machine.

The solution present in [56] also handles the case of sending an email content to multiple receivers. The case of caching email content that waits to be pulled is solved through assigning peer groups for the replication of outbox area within the selected peers. By assigning a peer group for replicating the outgoing email content, the sender must not always remain online, waiting for the pull process to begin.

This approach represents one of the best solutions of handling email operations in a P2P environment. Although every process is clearly explained, this approach lacks in algorithms of carefully selecting peers across which the replication process occurs. By this I refer to solutions based on monitoring the activity of every peer in the P2P overlay and carefully selecting the nodes with above average uptime resources.

5. Interoperability Solution between current Mailing Systems

This chapter presents the work available in [59] where a reliable system interface between several Peer-to-Peer (P2P) mailing systems and current client-server based mailing solutions is developed. Several issues are raised by such an interface, one being that traditional P2P mailing systems need to handle internal protocols by biding to a certain RFC standard format. Another issue lies in the way peers are referred to from the outside network. This approach involves separating the RFC standard from the internal communication protocol between peers, thus enabling the interoperability between systems even if the RFC standard is updated.

Many of the current P2P mailing concepts treat their internal protocol as being compliant to a certain RFC protocol format. Due to this issue, it is very hard to maintain the compatibility with the newest update when handling the internal protocol directly and strictly according to a standard. In our days client/server (CS) based mailing systems are more commonly used instead of P2P solutions. This chapter presents a theoretical system interface to make the currently available P2P mailing solutions compatible with the ones commonly used. Adding interoperability between P2P and CS-based mailing systems holds potential benefits to both parties, through compatibility between two different solutions.

Current P2P mailing solutions were developed on both structured and unstructured overlay network. All the P2P mailing systems were designed to be compliant to the actual server-centric model according to a certain RFC protocol format.

The mailing architecture present in [54] was developed on a hybrid P2P network concept. It offers the possibility of grouping peers according to the region specification. Their internal protocol is handled according to another concept than the one used in the server-centric model.

The solution presented in [55] was developed over the Chord overlay network [19]. The authors also used a self-developed protocol for interaction among peers that contribute to the mailing system. They provided an abstract solution of an interface to handle the compatibility between their solution and the actual today's mailing implementation.

The solution found in [56] was developed above the framework provided by the structured overlay network concept. It offers a pull-based manner in retrieving the email content from the peers that belong to the same community validation. Also the communication relies on a self developed protocol, through which peers are handled into the same application layer: the mailing system.

In [57], all the elements that help shaping the mailing system are implemented in an object oriented environment and the communication between mailing parties relies on a self-developed protocol. The authors conclude their work with requesting an interface design that would enable the compatibility between their implementation design and the available email clients (e.g. Outlook) and the server-centric mailing system implementation.

Through the provided analysis on the previous work on P2P mailing concepts, the authors from [59] suggested that is necessary to design an interface compliant with the traditional mailing design (based on the server-centric model) and also with the commonly used mail client applications. The interface is built in the manner of splitting the connections used for the RFC standard format from the internal P2P communications. Also the shared space through which each peer

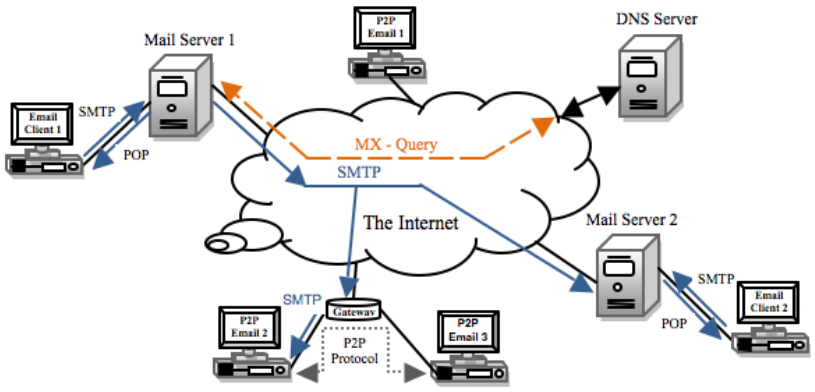


Figure 5.1 Interoperability Scenario between P2P and CS-based Mailing Systems

contributes to the mailing system is considered at an abstract level for providing a reliable foundation for the P2P application concept. In this manner the future implementations of P2P mailing implementations will also rely on a self-developed protocol format without the concern of inter-compatibility with the traditional mailing concept based on a server-centric model.

5.1. Architecture Preliminaries

The mailing system represents a complex infrastructure of a series of precisely aimed tasks. Figure 5.1 shows a possible scenario for interconnectivity and communication within a mailing system. For an ease of understanding all the components that help clarify the mailing tasks were represented outside the Internet cloud. Naturally, one email provider has its data centers distributed according to geographical distribution (e.g. google.com [60]) to assure certain agreements, such as: service uptime, store and data availability, service performance, etc. In this example elements from different Internet service providers (ISPs) were used to illustrate the mechanism of interoperability.

As previously mentioned before, the task of sending/receiving an email content is fulfilled at an abstract level by both the mailing client and the server side. If user 1, that uses the traditional mailing system, wants to send an email to user 2 from the same mailing service type, its mail client application contacts first the assigned mailing server for that purpose. The mailing server performs an mx-lookup to retrieve the mx-records from the domain name system (DNS) service [52] according to which it finds out the user 2 destination server. Usually the requesting mail server takes the mx-entry with the highest priority and tries to establish a connection with the user 2 receiving server. After the connection was established according to the mx-entries, the server handling user 1 email request, sends the content via SMTP protocol to the server where user 2 is usually connecting and performing his daily mailing activities. When user 2 wants to read its emails, it connects to the dedicated mailing server and retrieves the new mails via the POP protocol.

The P2P mailing infrastructure still represents a new concept over the network infrastructure, and because of the behaviour of its peer members (uptime is

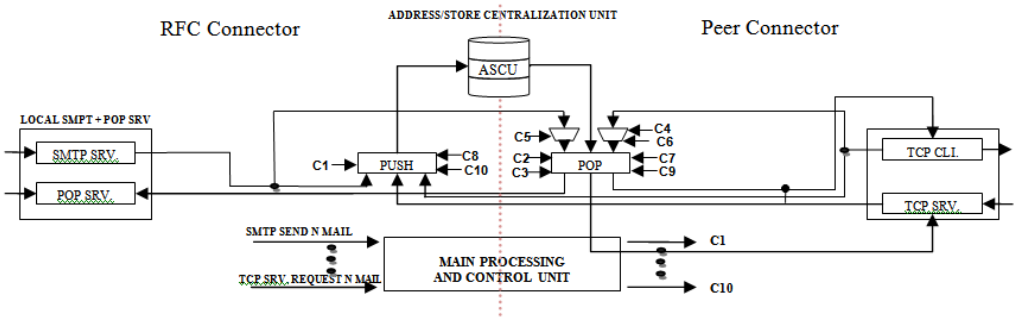


Figure 5.2 Interoperability Interface between P2P and CS-based Mailing Systems

unpredictable) it is very hard to determine a fixed address of such entities. Therefore, a situation is presented in which the P2P mailing concept is present in an institution [55] situated behind a gateway with a fixed address (IP address). For an inter-compatibility with the traditional mailing system, a few number of peers must be registered to the DNS service as mx-hosts, and also an implementation of an SMTP interface is required. Hence, when the mailing process takes place from a traditional mailing system to a P2P infrastructure or reversed, the same steps are performed: retrieving first the mx-records, establish the connection with the mx-host and perform the sending process of email content. When User 1, which uses the P2P mailing infrastructure, is registered as an mx-host, receives an email with the destination User 2 from the same mailing service type, it notifies the destination user for new mail notification (if the notify feature is implemented). User 2, than retrieves, according to the internal P2P mailing protocol, the new email content.

5.2. Architecture Implementation

The interoperability solution between P2P and server-centric mailing systems relies on the interface presented in Figure 5.2. The interface separates the protocol used under the RFC standard format from the internal protocol of inter-peer communication. The RFC connector is used mainly for translating email content from one side to another (P2P to/from RFC protocol) and for providing compatibility with the traditional mailing system. The work in [57] suggests that the P2P email content should travel according to a self-developed protocol, for separating the RFC standard from the internal P2P communication. Only solutions for the TCP/IP network layer were provided, however, for any other protocol implementations, which are positioned higher or lower than the one presented in this chapter, the main process remains the same.

For handling the disk space every peer is willing to share, the necessary connections were provided to all the elements that help handle the interface design. The shared space, specified here as the address store centralization unit (ASCU), is protected against concurrent writing, through the presence of both reading and writing buffers (POP and PUSH). Because the interface is implemented as an additional application which serves as a service for the users of P2P mailing infrastructure, it acts as a process (main processing and control unit) that handles the internal blocks through several operational tasks (processing threads) activated by the signals present in Figure 5.2.

5.2.1. RFC Connector

The RFC connector specifies both the SMTP and POP connectors used for communication with the traditional mailing system. The SMTP server interface is used for receiving email messages content in an RFC standard format. If the peer is registered as an mx-host, the SMTP interface binds to the assigned gateway address, otherwise it uses the local host address (IP 127.0.0.1) only for mail client connection purpose. When data is to be sent to this interface, signal c1 notifies the request of storing data to the ASCU through the PUSH buffer. Depending on P2P email protocol, the data newly arrived through this interface is automatically adapted to the one used internally by the mailing system. If the destination of newly arrived email represents the same peer host address, the email content is also available on the POP buffer through signal c5 notification, if the mail client application is also connected to the RFC Connector (POP server interface).

The POP protocol is used for retrieving email content from the ASCU after the authorization process of a certain mail client request. This interface only binds to a local host address and its main purpose is to answer to the mail client application request of retrieving new incoming mails. When the mail client requests the incoming mails, the POP interface signals the ASCU to make the new emails available on the POP buffer (c2 signal). Through the RFC connector only the case of store and forward mailing system property was handled, hence when an email is retrieved through the POP interface signal c3 is also generated and its presence tells the ASCU that the emails that are being retrieved from the mail client application side are also marked for deletion from the shared space.

5.2.2. Peer Connector

There are several platforms through which the P2P mailing systems have been developed and improved. The Peer Connector is considered as an abstract solution for either the hybrid or overlay platform implementation of mailing infrastructure. Either the implementations, the Peer Connector must consider both the shared space and RFC Connector as independent entities for maintaining the compatibility and format according to the server-centric mailing design. The TCP/IP network layer is considered as being the foundation of other protocols developed higher or lower than the one mentioned in this paper work.

The TCP Client and Server perform two different tasks: intercommunication between peers according to the used mailing architecture design and the notifications used for sending/retrieving email content. The work in [59] handles the case scenario of one peer lying behind a network address translation (NAT) server, which combines firewalls and dynamic IPs for blocking connections inside the protected network. In this case both the TCP Client and Server have the property of retrieving and sending email content in a direct relation with the ASCU and RFC connector entities.

When the TCP Client receives a new email content it notifies the POP interface through signal c4 if the receiver mail address matches the peer who handles this operation; or thorough signal c8 for storing the email content for other peers that are not online or have not read their email for a while. For retrieving an email according to the internal P2P mailing protocol, the client signals the ASCU

through the c7 signal for having the data available in the POP buffer for transmission.

The TCP Server performs the same operations as the client: it notifies the POP interface through signal c6 when the email has reached its destination, it stores another peer's email content according to signal c10 and is ready for sending cached email content to another peer destination under the c9 signal.

5.2.3. Address Store Centralization Unit

The P2P concept implies most of the cases that participants contribute, besides the computational power, with a certain percentage of disk space. Regardless the operating system or carrier (mobile/desktop), the shared space must be considered as a protected entity against failures that affect both the consistency and privacy of data. Although for a mailing system, each email content is protected according to the PGP (pretty good privacy) [61] method, data consistency should also be consider as a reliable way of handling data integrity. The ASCU entity is handled as abstract in the work available in [59], because every P2P mailing implementation comes with a self-developed protocol through which data is also being handled according to a different format.

5.3. Conclusions and Discussions

The authors from the work available in [59] solved the issues raised from the interoperability request between the P2P and client-server mailing architectures. They have provided an abstract model of an interface through which solutions of handling both the internal peer-to-peer and server-centric communication protocols were shown. This work also pointed out the cases according to which our interface model represents a good solution for handling inter-compatibility between the two mentioned concepts. The solution has been also tested with a self-developed P2P mailing infrastructure, designed over a hybrid platform.

6. Improving P2P Mailing Architecture mechanisms

This chapter handles the software implementations of mailing architectures across the Peer-to-Peer network environment. Although current implementations harness the benefits of such distributed environments, there are still research fields that can contribute significantly to their improvements:

- **Dynamic Environment**. Both structured and unstructured infrastructures were designated for implementing mailing architectures.

- **Replica Placement Algorithm**. Replication techniques are implemented on a P2P environment with an uptime expectation clearly specified.

- **Security**. Provided through encrypting data according to the public/private key pair.

- **Heterogeneity Classification**. The backbone of P2P mailing operations rely on nodes with above average resources such as CPU power, increased uptime and bandwidth, and shared space.

- **Interoperability Issues.** The proposed mailing architectures should be inter-compatible with the ones available today (traditional) in terms of the following operations: send, receiving and notifications.

Chapter three provided a thorough analysis regarding the P2P architectural design implementation. Statistics results have shown that promoting nodes with above average resource expectation generates a reliable environment across which the backbone of any application can rely on. Through designing a P2P mailing architecture, the following computing resources were carefully chosen to validate the classification among ordinary nodes and super nodes: computing power, uptime evaluation, increased bandwidth expectation and shared space.

Throughout the developing of any of the presented mailing architectures DMS [61], HMail [63] and DMail [64], which represent the authentic and innovative part of this thesis, attempts of harnessing computing resources across P2P overlay environment were made in several directions. One direction focuses on peer resources already evaluated as super nodes within the P2P network. Through this method of handling peers directly according to their SN state, structured and unstructured architectures were chosen to rely their backbone on such nodes: Two-Tier Chord extension [36] and the multi-ring topology available in [62]. In the mailing architecture present in [64] built over the Two-Tier extension of Chord, operations are divided according to the classification of joining nodes: caching and replicating email content occurs on ordinary nodes and operations of sending, receiving and notifying occur on the super node side.

The mailing implementation developed in [61] has an implementation related to the validation of super nodes within the P2P overlay. Through the use of multi-ring topology available in [62], an architectural design of three tiers was built. The first tier is used for caching and replicating email content across ordinary nodes. Both the second and third tiers are used for building communities in a ring shaped

manner for providing inter-communication between communities and email operations. The inter-communication is handled internal (marked by the same geographical area distribution) and external (third tier) by handling communication between communities from different geographical areas.

A second direction of handling peers heterogeneity is present in the work available in [41] which places resources of joining peers gradually on several hierarchical layers. The same community validation of geographical distribution is implemented in HMail [63], each community being defined as a hierarchical module composed of two tier layers. Both layers define an inter-connected ring topology built according to the Chord protocol specification [19]. The first layer validates node with above average bandwidth and uptime resources and the second one validates nodes with an increased status of shared space and computing power. The content caching and replication occur on the second layer tier, this layer level providing also security facilities in terms of unauthorized aces from other parties within the network.

A thorough analysis of uptime characteristics of peers within several different implementations was provided in Chapter three. The peer availability within file sharing and VOIP applications developed across P2P overlay networks was analysed. As a conclusion, the applications based on facilitating only communication presented the best results in terms of peer availability. The mailing application is also a communication based application that everyone has adopted as daily based tool. The average uptime expectation of super nodes within the Skype implementation reached 5.5 hours daily. Although the email content replication occurs according such premises of high peer availability, scenarios ranging from worst case (0.1) to best case (0.9) were also considered. At the expectation of 0.1, an email availability of several minutes, distributed in a time interval of 24 hours was considered for replicating the email content. Through the expectation of 0.9, peers with an average uptime status of 5.5 hours spent daily across the P2P overlay was considered. Nevertheless, the increased uptime expectation of nodes with above average computing resources, is used in handling the P2P environment across several hierarchical tiers for providing task balance among participants. Without the increased expectation of uptime facility, the backbone of such architectural implementations cannot be sustained from ordinary nodes, which are joining the network only for a small amount of time.

The P2P mailing architecture proposals are conform in terms of interoperability with the ones available today, based on server centric architecture. Two directions of compatibility were handled with the traditional implementations. The first direction handles only incoming mails from the server-centric architecture to the P2P mailing implementation. For this purpose the interface presented in Chapter four is required. Only the protocols of SMTP and POP were handled as traditional standards in handling email content across the network. A second direction is based on a hybrid solution, which combines both technologies (P2P and traditional) for an overall cost effective mailing architecture concept design. For this matter the P2P mailing mechanism must handle also outgoing email content according to the previously mentioned standards.

The security facility was addressed according to the public and private PGP [40] keys. To perform a fully secured communication, one could easily extend this security model by requiring the services of an external certificate authority, which could provide a higher level of security.

6.1. Distributed Mailing System (DMS)

The work available in [61] presents the concept of a new mailing infrastructure over an unstructured Peer-to-Peer overlay network, covering the system architecture and the specific mail operations. Current P2P mail implementations are built on top of lower layer network architecture called overlay, which is typically designed to rely on homogeneous computing resources such as: bandwidth, computing power, storage space, etc. This proposal classifies the participants to the network in *entities* and *super nodes*, depending on own resource evaluation methods. This enables each peer to effectively contribute to the mailing system according to the real performance of its resources, therefore increasing overall application performance and reliability.

The presented mailing architecture model uses the concepts found in [54, 55] and is developed over a hybrid P2P network design. The architecture is structured according to a community validation, each community being composed of several super nodes; each super node managing a limited number of other nodes called entity nodes. Each compound of the communities that address the same location identifier has a member in its community that is addressed by a server-centric element. If the destination of one's email receiver is out of the sender's super node range, it contacts the member community for querying the sender address. In this approach there is no need to rearrange mail data content over the nodes that are currently online. It uses a prediction method for synchronizing the data across entity nodes over a limited uptime interval.

6.1.1. Preliminary Assumptions

The architectural model used for designing this mailing system relies on the concept used in [62], describing an interconnected multi-ring topology (Figure 6.1). Each network ring model defines a community through interconnecting nodes that present higher system resources than ordinary network participants (entity nodes - E), called super nodes (SN). The interconnected rings are distributed over the network according to an external location service provided by an external location service such as MaxMind [42]. Through the GeoIp tagging, there can clearly be distinguished across the network which nodes should interconnect and which should not, according to the information provided by MaxMind: hostname, country code, country name, region, region name, city, area code, etc. To prevent unnecessary bandwidth usage, queries over the rings take place only by local area limitation (TTL - limited) described in [42]. To overcome those limitations, a dispatch ring community is present in every location area, being addressed and managed through an external service of domain name service (DNS) [52]. Hence, every query addressed outside the local area limitation is directed to the dispatch community, ensuring optimization of network usage.

Three types of network links are handling communication in this design: external, internal and local connections. The external links are used for interconnecting ring topologies, links sustained only from the super nodes participants. The internal links are used for connecting super nodes inside a ring and to lower the time needed for propagate a query inside the ring. And the local connections are held between entity nodes and super nodes for assuring load balance among email operations and maintaining the mail service alive.

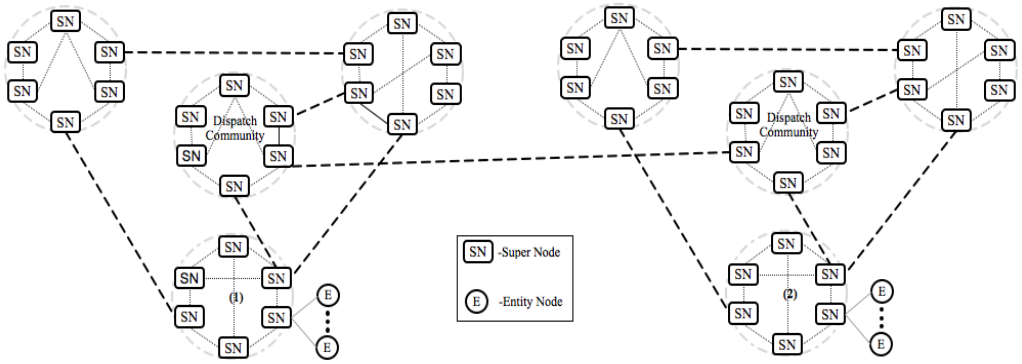


Figure 6.1 Distributed Mailing System Architecture Overvies

In Figure 6.1 the links used for interconnecting mail system parties to the network service are shown. The dashed lines represent the external links, the dotted ones the internal links and the ones with grey represent the local connections with entity nodes. Through the information provided by MaxMind yields the membership of one node to one certain country and the region inside that country. Because limited TTL broadcast messages are used in this mailing architecture, a limitation is imposed by the community connections to the same region code. Only dispatch communities can perform connections to other communities from different region codes, but under the same country code and only between dispatch communities. If one query aims higher than the country distance limitation, than it asks the dispatch community to address the destination address through the external service of DNS, to which all the dispatch units are registered with a limited number of super nodes members from each registered community.

Grouping unstable network parties together represents a major challenge for a system that is unstable itself. The factors used in deciding which participant to the mailing architecture gains the property of super node after a self evaluation process are bandwidth, uptime, shared space and computing power. For each of the considered metrics, evaluative score points are assigned, the final result being computed according to a weighted average formula.

Bandwidth (as a metric) is expressed in terms of download and upload speed. Typically, Internet service providers (ISPs) assure higher download speed than upload because they have designed their systems to optimize download speeds. Under these circumstances, in this architecture design the bandwidth measurement is evaluated according to a weighted average, the upload speed being offered a much higher weight.

Each participant should contribute to the mailing system by sharing some percentage of its disk size. The shared space represents a small piece of the system's database. The mailing system is designed according to the concept of a network attached storage (NAS), constructed from small sizes of disk spaces that each user is willing to share. Unlike the NAS architecture, where disk storage failure is controlled, watched and managed, DMS storage comes in a variable and uncontrollable way. The shared space structure remains abstract for this research.

Regarding the computational power, there are several systems that have different hardware configurations. Computing power will be tested in time, to see how a peer handles its participation to the system. This will test how many threads

a computing system can handle, access time to the local disk, memory availability, etc. Periodically the application tests the CPU workload and how much memory is required by the main application process.

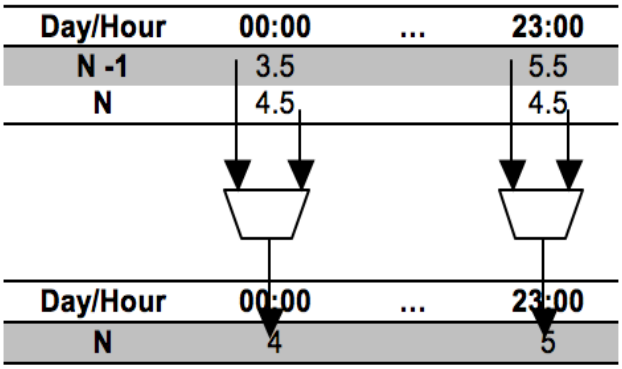


Figure 6.2 Uptime Prediction Evaluation

Uptime represents the key factor in data caching and replication. To provide a solid foundation for grouping participants, the mailing architecture is designed according to an uptime availability prediction. The concept found in [47] provides a thorough analysis of peer availability prediction over the network. The concept of the inspired work relies on the number of counts per time interval sent periodically from the peers that are still up in the network, letting the peer neighbours know the current state of uptime availability over a period of time. The count unit is measured according to a time slot of five minutes, generating a 12 time slots per hour, 288 time slots per day. The method used in [47] could generate a good prediction within an interval of a week.

The work available in [61] handles uptime availability as an average mean of a time slot of 60 minutes generating 24 time slots per day. The authors concern remains only to predict what are the chances that a peer is available on the network at a certain moment in time. In an unstable network architecture design, one cannot predict precisely the moment when a peer will be up and running. Hence, the focus is set on finding the total number of peers across a community needed for caching and replicating data across an interval of 24 hours time slots. The history background for providing a good analysis of uptime availability prediction is provided in 5 days of peer observation.

Figure 6.2 represents one example of this method of analysing uptime prediction of a certain peer. Assuming that the highest score point for a time slot of one hour is 10 (a full range of 60 min), at day N – 1, the peer has obtained the score of 3.5 at 00:00 AM and 5.5 at 23:00 PM. But the next day the same peer has gained the score of 4.5 for both the same time slots. A mean value is then computed and the final result remains 4 for the AM time slot and 5 for the PM time slot. The score points can vary according to the peer's contribution to the mailing system.

Also the work present in [47] provides an analysis of peer availability through the use of BitTorrent, an application which holds 53% of all P2P traffic on the internet. Measurements were taken at geographical distribution level, acording to MaxMind [42], yielding in 191 countrys tested with an average availability of 28.39%. The analyzed uptime availability was different for each timezone, fact that

provides a good foundation for grouping participants according to the location services for our implementation design.
The final score point evaluation is computed according to a weighted average, where the uptime has the greatest weight:

$$ScpE = \frac{Scp_{uptime} * 4 + 2 * Scp_{bandwidth} + 2 * Scp_{shared\_space} + 2 * Scp_{Computing\_power}}{10} \quad (6.1)$$

6.1.2. Service Primitives

Throughout the evolution of traditional mail protocol, the RFC standard format has permanently changed, from the beginning of the mailing service until today. To adapt constantly to the newest protocol available on the market and to assure compatibility with the traditional mailing systems, this architecture was designed to perform intercommunication between peers according to a self developed protocol, maintaining the RFC format as an interface between the MUA and every peer joining our architecture design. The concept was also used in [56], and the interface was built according to a local SMTP/IMAP server which kept the compatibility with the MUA client according to the newest RFC standard format.

Being able to perform the simplest mail operations (send and receive), several service primitives were developed to help building certain tasks such as: store, delete, fetch, append, read inbox and garbage collection. For security purposes, the external services of PGP keys [40] implementation are required, assuring this way data security and user privacy. Also all user IDs should append after the domain name, the country and region code according to the MaxMind external service, for ease of identifying users addresses among the communities that form our network architecture design (i.e. user_id@domain.contry_code.region_code).

Due to our three layered architecture design, a store primitive is defined for each of the following: dispatch community layer, community layer and entity layer.

Throughout the community layer, lower peer elements (communities or entities) are being managed. Hence, data availability and load – balance features are defined through the presence of the internal connections between super nodes. Every super node must replicate its data according to the prediction method presented in the previous section. Because the final computed score point highlights only peers with super node property among communities, a thorough evaluation is performed to fulfil the availability feature. Therefore, score points that represent the lowest unit (hour unit) are used. One super node must replicate its data according to a 24 hour score point interval. The process is performed randomly across the community, resulting the internal community links. Through the presence of the internal links, a two-sided load-balancing feature is gained, for data and communication. Load balancing is gained through caching replicas among a limited number of super nodes; and through lowering the time needed for a query to travel in a community.

The difference between dispatch and the lower layered community is the caching content and the amount and type of queries. The dispatch unit is the one who manages lower layered communities in the same region code provided by the external service of MaxMind [42]. Therefore, it only performs connections with the

lowered and other dispatch communities. The dispatch community manages information regarding the area region code for which it is responsible, concluded in: all user region IDs, public PGP keys and lists recording the community address for each registered user (limited number of super nodes). Also, the dispatch unit purpose is to perform load-balancing among queries aimed at the same level, and not to forward them to the lower communities.

Each lower layered community manages its information according to the present number of entity peers. Hence, information is distributed among community units, data replication occurring only inside communities and not between them. The information that resides in every community is concluded in: public PGP keys of each peer connected to the same community, individual lists of received mails, individual lists of peer score point evaluation (score point table - SPT), and individual lists of the last sent mail addresses in a MRU manner (most recently used). All the information is replicated among super nodes according to the method presented in the Preliminary Assumptions section. At this layer level, the super nodes interconnected with the scope of replicating data across the community, are also building lists with entity nodes addresses according to the 24 hour validation, forming the storage availability table (SAT). This is mainly done in the idle time, when no requests (or very rarely) of email operations take place. The storage table is used for replicating data among a limited entity peers that together provide a 24 hour data availability according to their score point evaluation.

The *delete primitive* is implemented according to each of the following layer's validation: entity node, community and dispatch community. The delete operation can be triggered from the user side through erasing email content by reading inbox (store and forward mailing property); or by the garbage collection primitive, when no activity from the peer side was registered for a period of time.

When the read inbox operation takes place, the user only requests the email content from few number of entity peers available on the network at a certain moment in time. During the download of email content, the sender peers automatically mark the sent item as ready for deletion. After the upload is completed, the entity peers delete the email that was earlier sent to its receiver and inform the community that the email was successfully sent to its destination. The community stores this information for signalling other entity peers, that shared the same sent email content, to delete this item from their shared space when logging into the mailing system.

The content of every email that was previously sent to its destination is stored among a few number of entity nodes managed from the community where the sender logs in. If the email marked as unread is never read by its receiver, it is automatically erased by both the community and the caching entity nodes sides. This is done through assigning both the header and email content with a number of counts (measured in days), that both community and entity nodes decrement, when a day passes by. When the number of counts reaches zero, the email content is automatically deleted.

At the dispatch community layer, information regarding the user and email inbox is handled. The inbox entries are marked also with the count of days. The super nodes being in charge of holding ones email inbox, browses daily the list marking each entry with a decrement of one, deleting also the entries that have reached zero value. Also, the group of super nodes being in charge of managing and building the SAT tables, are performing daily the validation of each peer score point evaluation. If no score point is registered according to one day validation, the final score point is computed with the average mean of zero value. When the final score

point of one peer is equal to zero, the peer is marked with a count of days. After the count reaches zero and the peer hasn't registered to the mail service, it is automatically deleted from the community database.

When a user is deleted from the community database, the unit in charge must announce its deletion from the dispatch community also. The dispatch community performs the deletion operations only at the same registered area region community that the dispatch community is currently managing.

The *fetch* and *append* primitives define the operations of sending and retrieving items through queries addressed among the peers that form our network architecture design.

The *fetch primitive* is used when information is required between communities with no specified destination address. Before replicating the email content among the entity peers specified through the SAT validation, the community must first know the receivers public PGP key, according to which encryption takes place; and the receivers community address (number of super nodes that handle the community, within the receiver logs in). The super node handling the sender, is verifying first the receiver's country and region code appended after the domain name, in the specified user id. If both the country and region code match the community's region code, the fetch operation takes place through queries addressed as broadcast messages. If one of the country or region code are different from the hosting community, the dispatch community is being addressed to forward the query. If the dispatch community is unreachable, the super node from the hosting community uses the external services of DNS, being able to reach one of the super nodes that form the dispatch community. Every query addressed outside the hosting community, contains in its header the sender address of the requesting super node. When the query reaches destination, the receiver can directly address the sender through the information specified by the header.

The *append primitive*, usually takes place after a fetch operation, with a well known destination address. After the process of replicating the encrypted email content on the entity nodes, the super node handling the sender connection is now appending the notification (with the addresses of entity nodes replicas) to the community where the receiver logs in.

The *read inbox primitive* occurs between the MUA client and the interface provided by the DMS service application. The interface is hosting the local SMTP/IMAP server, and communicates with the MUA according to a specified RFC protocol format. The authors [61] propose this implementation for an ease of updating the protocol according to the latest version available on the market. Hence, when an update is available for the RFC protocol, the architecture design requires updating a small amount of data for a better quality of service.

When the user downloads its email according to the list of received emails headers available on the hosting community, it also deletes the email content from the entity nodes. The mail service is implemented as being one of the store and forward service type. Therefore, the user's MUA is in charge of replicating downloaded email content on the computing machine that served as a peer in this architecture design.

6.1.3. Email Mechanism

For sending and receiving email content through our network architecture design, the primitives defined in the previous section are used. An example of the

email operations is presented through the architecture overview present in Figure 6.1. For further explanation, both the users computing machines are evaluated from the DMS system as being entity nodes handled from different community locations (same operations take place if the computing systems are evaluated as super nodes). The steps needed for sending an email *m* from sender S handled from community 1 to receiver R from community 2 are explained in the following:

1. The user sends his email via the MUA client that connects to the DMS interface application (local SMTP/IMAP server) by specifying in the sender field the receiver's R user ID, domain name, country code and region code (R_ID@domain.country_code.region_code).

2. The peer property of the sender's user machine, evaluated as an entity node, makes the request of sending email content to the upper community layer.

3. The super node from the community that currently handles S connections, verifies if the country and region code matches its current location. In this case only the region code is different from the current location, and the super node forwards the request as a fetch operation to the dispatch community.

4. The dispatch community receives the super node's fetch request and verifies if the specified location address is handled from one of the neighbor dispatch communities. If there are no links with the desired dispatch community, the fetch operation reaches its destination according to the external services of DNS. In this case the requested community is directly connected to the dispatch unit that matches the fetch operations request, and the query is forwarded to it.

5. The dispatch unit matching the same location ID as the receivers email address, responds to the super node that it initiated the fetch request operation with the community address that handles R connections and its public PGP key.

6. The super node receives the information according to the fetch operation, and responds to S with the public PGP key and a list of entity nodes according to the SAT evaluation for replication purposes.

7. S encrypts the email content according to R's public PGP key, and starts the replication operation with the entity nodes provided in the SAT table.

8. After the replication process, the super node appends the notify header to the community that currently manages R's connection. The header is also encrypted according to R's public key.

9. R performs the read inbox operation and downloads the email content from the entity nodes that currently are online in the network in the community where S has sent the email content.

6.1.4. Experimental Results

The mailing system was simulated in an object-oriented environment, where both the entity and super node were handled as objects. The simulation was performed closer to the research provided in [47], through which the users are being characterized though their behaviour in time spent over the Internet. Hence, an analysis was provided of the possibilities ranging from the user who only remains logged in to he mailing service until it finishes reading emails (0.1 probability), to the user with an increased spent uptime (0.9 probability).

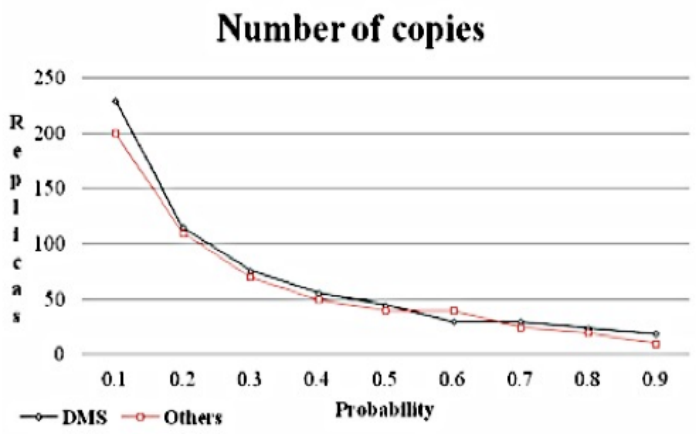


Figure 6.3 Number Of Email Replicas

Figure 6.3 illustrates the number of replicas, of one individual email sent from the user to destination, over a number of entity nodes designated from the super node, handling the senders connections, according to the SAT probability prediction. The obtained results are higher in number of replicas than the other previously researched solutions. The authors aimed in obtaining solutions precisely for situations in which all the participants act according to the uptime probability described in Figure 6.3. The worst case scenario represents the case within the participants are contributing to the mailing system according to the probability of 0.1. Because in this case users log in to the mailing system only for fetching email content, it is very hard to predict the moment according to which the same process can take place the day after.

Figure 6.4 presents the availability analyses for the results obtained in Figure 6.3 according to a time window of 31 days. To reach in practice, the expected results can be interpreted at the probability corresponding to the 0.5 – 0.7 range of values. That is because the users cannot be described according to one category of uptime probability. Hence, a good foundation for data availability under variable circumstances was provided.

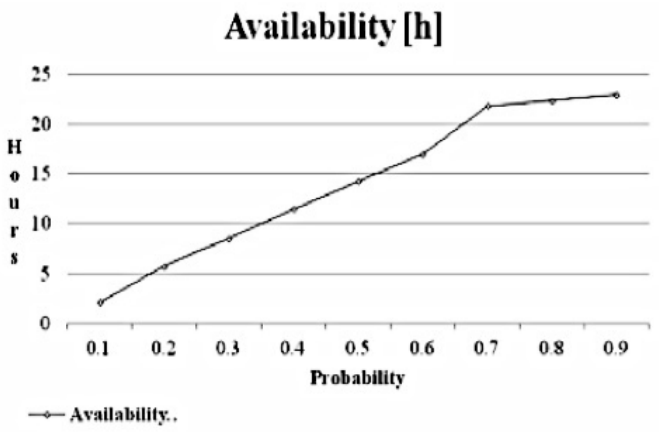


Figure 6.4 Average Email Availability per Day

Figure 6.5 represents the bandwidth according to which email content can be downloaded from a limited number of entity nodes from the receiver side. The variation of speed for different cases of uptime probability marks the fact that the mailing system relies mostly on uptime requirements than the bandwidth properties of a certain entity node upon deciding the number of peers according to which replication can take place.

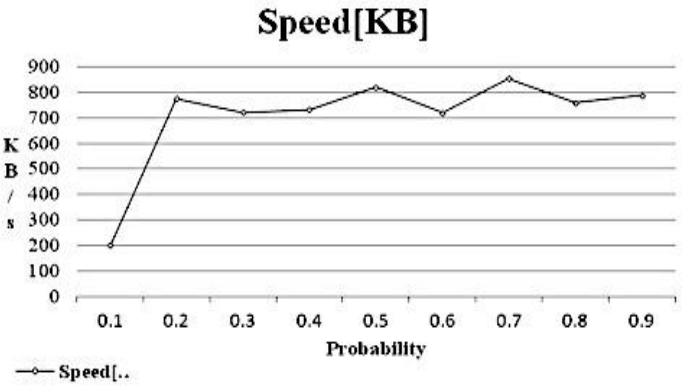


Figure 6.5 Download Speed

This work presented a new concept regarding the mailing infrastructure over an unstructured peer – to – peer network. A model of interconnecting peers according to the location services and dividing them according to the user behaviour in time spent over the Internet was shown. Also, a thorough analysis was provided regarding the uptime prediction according to which data caching can take place at any peer with a regular defined custom in terms of uptime availability. The obtained results show that even users with low uptime probability can be used as targets for caching data, but with an increased cost of higher number of replicas of email content.

6.2. HMail: A hybrid mailing system

Current mailing systems have adopted a server-centric model in handling email traffic over the Internet. Although the traditional mailing providers employ a large number of servers where mail operations are evenly distributed, all the emails are routed to a central gateway, resulting in accessibility issues if the gateway link is severed. Moreover, the necessity of having dedicated buildings and trained personnel for handling large email operations and network traffic is unavoidable.

The work present in [63] introduces a new mailing architecture design by combining the research activities on developing a hierarchical Peer-to-Peer (P2P) framework [41] and resource evaluation methods of a certain peer in the network. Users are grouped according to their geographical distribution for assuring a load-balance in network traffic regarding email operations and present a way that helps the traditional mailing architecture rely on certain Peer-to-Peer decision blocks.

The author's contribution to the existing P2P mailing architecture consists in separating the decision tasks that facilitate the mailing operations over several hierarchical blocks distributed geographically. For this purpose one can specify on which computing resources the mailing operations can take place, and further, the solution can provide a better security of data regarding the email content, storing, sending and receiving operations. The authors chose to classify the hierarchical layers by the geographical criteria of peers because of the P2P network behaviour (instability of network). It is much faster to retrieve information from a peer located geographically closer to the requester.

6.2.1. Architecture Preliminaries

HMail infrastructure relies on a hierarchical extension [41] of the Chord protocol [19]. The hierarchical extension over the DHT framework provides some additional features for the flat overlays like Chord regarding the P2P application layer. Although the Chord overlay treats peers as being homogeneous resources in the network, the solution found in [41] suggests that according to each P2P application requirements specification, several hierarchical layers should be built. The application requirements can be seen as computing resources of the participants to the P2P network, such as uptime, computing power, bandwidth, shared space, etc. The hierarchical layers built on these requirements are handled into restrained entities over Chord, called hierarchical modules (HMs). Every layer is built according to the Chord protocol by using the same nodes from the overlay. Each HM is managed through a control file (dispatch list) stored on the Chord overlay, described here as Base Chord Overlay. The lifetime of a dispatch list is limited, determined by the valid entries of certain peers that are carefully selected according to the application requirements in handling certain hierarchical layers of the HM. Each control file has two sections: general set of rules, defined from the upper application layer, which specifies the application requirements for a certain HM; and entry points for each set of rules, where information about few number of peers is registered for each hierarchical layer (IP address, Port number, login time).

There are two major structural components that build the HMail mailing architecture design: the overlay framework and the application layer. The mailing application layer basically performs three tasks: retrieve email from sender, store the email content and send email to receiver when queried (store and forward). To be able to perform these tasks safely on the P2P environment, where the behaviour

of participants is very unstable, the application layer should refer only to the nodes that meet higher requirements in terms of uptime, bandwidth, computing power and shared space. In the previous research [61], the authors have promoted super nodes for this matter (peers with more than average computing resources), but the expectation ratio of nodes performing all these requirements together was low. Therefore this architecture design will take in consideration nodes that meet some of the desired application requirements gradually on hierarchical layers above the overlay framework. For this purpose the concepts used in the research found in [61] for describing the terms of joining one hierarchical module of the overlay are used.

A major challenge in handling participants over a P2P network architecture is finding one support group of nodes that can perform a variety of operations in a stable manner. Finding such a support group of interest implies also fulfilling the requirements expectation for a certain application. For the HMail architecture design the authors from [63] require that participants meet the following resources expectation: uptime, bandwidth, computing power and shared space.

The uptime prediction method remains one of the most critical requirements in building the mailing application. Because of the unstable P2P environment, no algorithm can exactly predict the moment when a peer will become active in the network. The algorithm found in [61] is used for this matter, which associates every node in the network with its own uptime evaluation in terms of score points generated across an interval of 5 days of history analyses.

Because Internet Service Providers (ISPs) provide uneven bandwidth resources concerning the upload and download speed, the requirement expectation regarding this resource is focused more on the upload speed. Nodes are grouped according to this metric for providing quality of service regarding the email operations for the HMail architecture design (sending/receiving).

The computing power designates the quantity of operations a certain peer can handle. Through quantity, the number of processes is referred that a peer (as a computational machine) can handle at a certain moment in time, memory availability, access time to the local disk, etc. Only nodes with this feature can reach upper layers in the hierarchical module above the overlay.

The shared space represents the storage space assigned to the mailing system, composed by the amount of space given freely from the peer side.

6.2.2. Basic Components

HMail platform framework design relies on several hierarchical modules built above the Chord overlay (Figure 6.6). Every HM identifies itself through the geographical area distribution of its nodes (peers). For this purpose the service of an additional application is required, such as MaxMind [42], which provides the following information based on IP address of joining peers to the mailing architecture: hostname, country code, country name, region name, city, area code, etc. Every control file, describing a certain HM, is stored on the Base Chord Overlay under the hash ID of a certain country code. Therefore all user IDs should append the country code next to the domain name (ex. user_id@domain.country_code).

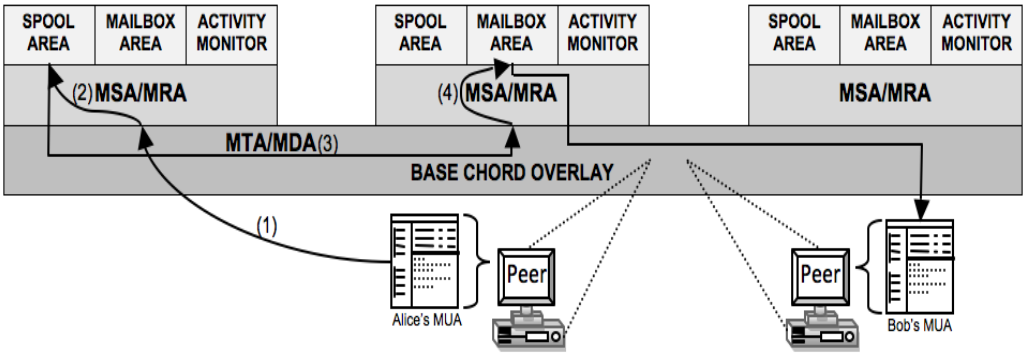


Figure 6.6 HMail Architecture Design

This way the application side can easily mark the membership of a certain user when email operations take place (joining, sending / retrieving messages, etc.).

The criteria in building the first hierarchical layer above the Chord overlay lies in selecting nodes with more than average uptime and bandwidth resources validated by the GeoIP tagging. This single partitioned layer defines the mail submission agent (MSA) and the mail retrieval agent (MRA) components of the mailing system. These components are used for sending and retrieving emails, and common overlay nodes (peers) that were not validated by the HM, can query only this layer of hierarchy.

The second hierarchical layer of a certain HM is built according to the validation of nodes within the first layer with more than average resources such as computing power and shared space. Only nodes with higher computing power can build the second layer of hierarchy at the same time as they build also the first one. Because all information regarding the mail operations and HM maintenance is stored at this layer, the validated nodes should also share some percentage of their storage space. This layer is split into three partitions: spool area, inbox area and monitor activity. The spool area represents a temporary holder for new emails placed from the MSA side in order to be sent to their destination. Also the nodes from this area perform the mail delivery agent (MDA) and mail transfer agent (MTA) components of the mailing system. The inbox area is the holder for all incoming mails referring the users with the same GeoIP tagging.

The monitor area represents the holder of public PGP keys [40] of mail users from the same GeoIP tagging and users from other HMs that are authenticated to place new incoming mail to the inbox area. As the number of mailing participants increases, the higher the HM demands. The monitor area also contains data for managing the HM, in terms of raising the expectation ratio of more than average resources, supplying the HM with more nodes when needed.

6.2.3. Email Operations

Through the proposed hierarchical architecture design, communication between mail components occurs in a restrained manner. Common mail users have access through their email application only to the first level of hierarchy of a certain HM. Although every node in the overlay has access to the information provided by any control file of a certain HM, the interaction between parties occurs according to

an authentication method based on PGP keys. To perform a fully secured communication, one could easily extend the proposed security model by requiring the services of an external certificate authority, which could provide a higher level of security.

HMail mailing application is a daemon that acts as a local server for common MUA client applications (ex. Outlook, Mozilla Thunderbird, etc.). It provides the protocol interfaces for SMTP (simple mail transfer protocol) and POP (post office protocol) according to a RFC standard format [49].

The authors [63] have identified three types of email operations on HMail mailing infrastructure: sending, receiving and deleting of email content. In these three basic categories, details regarding the store, authentication and garbage collector operations are specified.

In Figure 6.6 Alice's computer represents one of the member peers of the HMail architecture design. Alice uses her common email client to write an email to Bob (*sending operation*). When the email is sent from the MUA side, the P2P mailing application receives the email according to an RFC standard format and converts it to an internal one. Assuming that Alice's computing machine is not validated by an HM, the P2P application first searches the Base Chord Overlay for the control file of the HM who's ID matches the country code where Alice lives. With the obtained information from the control file, the application connects to the MSA layer of the queried HM. The MSA performs an authentication process according to the information from the second layer provided by the monitor area partition.

The MSA generates a random data and encrypts it according to Alice's public PGP key and sends it to the querying application. The application receives the encrypted data, decrypts it according to the private key, which is securely stored on Alice's computer, and sends it back to the MSA. If the MSA confirms the match, it requests for the email from Alice's peer side. When the process of email retrieving finishes, the MSA places the email content into the spool area partition from the second layer of the same HM.

The email placed from the MSA side is stored into the spool area under the hash ID obtained by combining the date, time and destination address. The nodes from the spool area are responsible for placing Alice's email into Bob's HM inbox area.

The node from the spool area responsible for holding Alice's sent email performs the operations described by a mail transfer agent. The MTA searches the HM identified by the country code appended next to Bob's email address. After fetching the control file from the Base Chord Overlay, the MTA connects directly to the inbox area of the HM where Bob logs in for checking new email notifications.

If no connection can be established to the inbox area, the MTA hashes Alice's email request by combining the date and time of the first attempt in connecting to Bob's HM, the number of connection attempts and the destination email address, and stores it back in the spool area. The period of time needed for requesting the same connection to Bob's HM is computed according to the number of attempts found under the ID of Alice's email request. If this number of attempts reaches a value specified in the monitor area, and no connection happened to be established to Bob's HM, Alice's email is stored in her inbox area with the appended message of error in sending the email request.

If the connection to Bob's inbox area is established, the same authentication method is performed through hashing a random data with the public PGP key of the node registered as an MTA to Bob's HM stored in the monitor area. After the

authentication has been succeeded, the MTA places the email on Bob's inbox area (MDA).

When Bob wants to check his inbox (*receiving operation*), he performs this operation through the MUA application. The request is sent through the email client by connecting locally to the mailing daemon. The mailing daemon connects to the HM identified by hashing the country code appended next to Bob's email address. After a connection has been successfully established with the mail retrieval agent, an authentication process is required in order to have access to the new incoming emails. After the authentication has been made, Bob's mailing daemon retrieves the new incoming mails and marks them as read. After fetching the new emails, the mailing daemon passes them the Bob's MUA according to the standard RFC POP protocol.

Bob's inbox holder is composed from an index file and the email messages that are stored separately on the inbox area. The index file is structured according to the new and old entries of received emails. Every entry specifies a stored email in the inbox area by appending next to the email header the hash ID of content in the inbox area. Also the index file contains information about the space used by Bob and notifies the daemon about the available store space.

The *delete operation* occurs due to mailbox area notification stored in Bob's inbox (index file). If Bob has exceeded the available storage space on the HM, first a notification is sent to the mailing daemon and Bob needs to perform the deleting operation himself. If Bob does not take action on the space notifications arrived from the mailing daemon, than the garbage collector automatically performs the deletion process. The garbage collector is implemented in an FIFO manner (first in first out). Only the oldest messages on Bob's inbox will be selected for the deletion process.

The garbage collector is implemented on every node from the inbox area of a certain HM. Every email newly arrived in this area is marked with a number of counts measured in days. Every day passes by the count number is decremented by every node that stores an index file in this area. When the count reaches zero value a notification is automatically generated on the inbox holder for notifying the user to take action. If the count reaches a negative value and the storage space limit was exceeded, the emails with the higher number of negative counts will be automatically deleted.

6.2.4. Interoperability Solutions

The authors from [63] have designed the mailing architecture according to the interface solution found in [59]. This way the P2P application runs as a service behind the operating system providing connectors for an RFC standard format (POP for retrieving and SMTP for transport email content). The authors considered two scenarios in handling operations with the traditional mailing service: standalone and hybrid mailing service.

When considered as a standalone P2P mailing application, the HMail solution design must handle only incoming mails from the traditional architecture. For this purpose a number of few nodes are required from the spool area of certain defined HMs registered as MX-Hosts in the domain name system (DNS) [52] named after the architecture solution: HMail.com. This requires that the nodes registered, as MX-Hosts should bind their SMTP connector to an IP associated with the HMail domain. When an email is sent to this architecture from the traditional service, the HMail

domain system is queried for retrieving the MX-hosts and emails are sent via SMTP protocol to the spool area of certain HM. If the email is intended for the HM available as MX-Host, then the spool area transfers the email into the inbox area of the recipient. Otherwise the spool area is in charge of securely transferring the email content to its destination.

A hybrid model of this mailing architecture design is based on the collaboration between the P2P and Client/Server structure model. The HMail domain includes MX-entries for both the adopted models: datacenter gateways for the traditional one and spool area hosts for the P2P architecture. When an email is to be sent to this architecture design, the designated MTA retrieves the MX-hosts for this purpose from the DNS. It receives a lists of hosts prioritized according to the P2P-traditional order. If the nodes from the spool area are not reachable, overwhelmed in terms of storage space, bandwidth usage, computational power, etc., the MTA choses the hosts from the traditional model. This way a focus us set more on the P2P model where the costs needed for handling the mailing operations are reduced than using the traditional one, where extra storage space involves new investments in terms of computing resources, specialized personal and dedicated buildings.

When an email is to be retrieved from the HMail architecture design a mail relay should be designated for this purpose. The mail relay can point to both P2P and traditional data holders according to the used location for storing the email content.

6.2.5. Simulation and Experimental Results

We HMail architecture design was simulated in an object oriented environment, where peers have been represented as objects within the application. Every participant to the network was simulated through the presented metrics, and every HM was represented through an additional object where peers can subscribe or leave according to their own resource evaluation method [61]. Because peers can join or leave the network at anytime, the authors [63] have also simulated case scenarios for each of the uptime probabilities: 0.1 for worst case scenario, where the entire network simulation depends on peers that join the network only for a short period of time to check their mail inbox status; and 0.9 where peers remain connected to the network for several hours daily.

In Figure 6.7 the number of email replicas needed for assuring the data availability on the P2P network is represented. The worst case scenario (0.1 of uptime expectation) requires more replicas than the other P2P mailing implementations [61][55]. Because of the unstable environment generated from peers that join the network only for a short period of time, it is difficult to maintain the HM structures above the Chord overlay and also to assure the availability of email content on the second hierarchical layer. As the uptime probability increases, the network environment becomes more stable and the number of email replicas decreases substantially. Because only the nodes with the highest resource evaluation can be part of certain HM resource, data availability is assured from a number of nodes proportional with the uptime expectation of the validated peers.

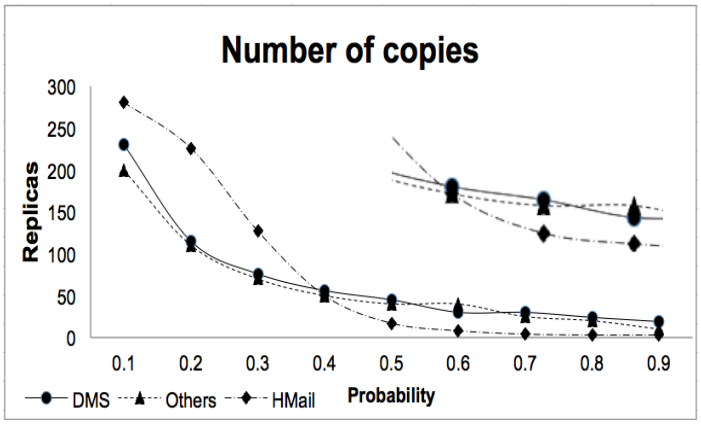


Figure 6.7 Number of Email Replicas, 1000 Nodes Simulated

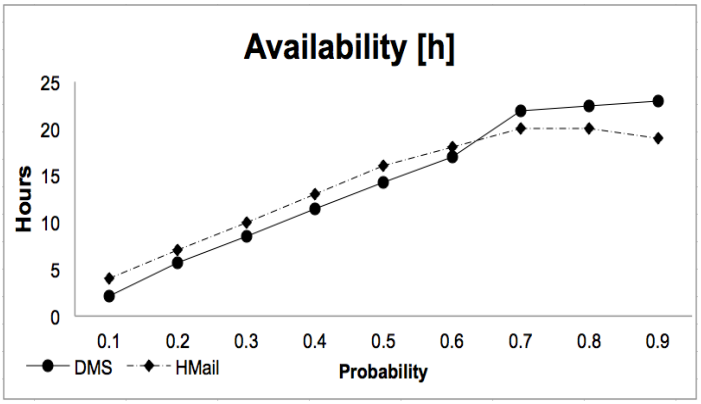


Figure 6.8 Average Email Availability per Day

In Figure 6.8, the authors [63] have represented the data availability expectation for each considered uptime case scenario and compared their results with [61]. For the worst case scenario, the email content is available on the mailbox area only for a short period of time that ranges from 5 to 6 hours per day. As the uptime expectation rises for the considered simulation environment, the mailing architecture design provides a stable data holder for the email content. The time interval needed for storing the email content for higher uptime simulation scenarios decreases because of the high peer resource expectation validated by the HM. When the resource expectations are high for a certain HM, it becomes difficult for a node to re-join a certain hierarchical layer. This is possible due to joining time offsets of certain peer in the network, which sometimes yields with a lower resource evaluation than expected.

To reach in practice, the obtained results ranged from 0.4 to 0.7 of uptime probability, are to be considered. This covers the case scenario of all the discussed user types.

6.3. DMail: Distributed mailing system

Current Peer-to-Peer (P2P) mailing infrastructures were developed on the foundations of unstructured and structured overlay models. The mailing systems developed under the unstructured P2P network overlay have promoted nodes with higher resources (in terms of bandwidth, computing power, shared space, etc.) as super nodes, attempting to centralize and concentrate email operations across stable peers. In the structured model, nodes have been treated as homogeneous resources across the network, and email operations were possible due to the complex protocol used for linking nodes in the network. The work available in [64] proposes a new model of mailing architecture developed over a P2P overlay network which combines the strengths of both structured and unstructured overlay frameworks by promoting super nodes as gateways across the main overlay. Through this implementation the authors provide load balance properties in terms of email operations, bandwidth usage and processing power among peers. The authors have designed the service as an integration model with the traditional Client/Server mailing architecture and an applicability solution for this concept model is presented.

The mailing architecture [64] design relies on the framework provided by the two tier overlay implementation present in [36]. The author's contribution to the existing P2P mailing architectures consists in separating the decision tasks that facilitate the mailing operations according to the geoIP tagging [42], providing load balance in terms of bandwidth usage and processing power among participants to our application. Further in this work, the authors provided an integration method with the existing mailing service based on the Client/Server model and a case scenario of applicability for this concept design.

6.3.1. Architecture Preliminaries

DMail mailing application is represented through a daemon that acts as a local server for common MUA client applications (ex. Outlook, Mozilla Thunderbird, etc.). It provides the protocol interfaces for SMTP (simple mail transfer protocol) and POP (post office protocol) according to a RFC standard format [49]. For security purposes, the external services of PGP keys [40] implementation are required, assuring data security and user privacy. To perform a fully secured communication, one could easily extend this security model by requiring the services of an external certificate authority, which could provide a higher level of security [55].

The authors [64] designed their mailing architecture according to the facilities provided by the two tier overlay concept [36]. Trough this option of splitting super nodes from others, the authors were able to filter decision tasks of certain mail operations.

In this model concept every lower tier of Chord implementation should be identified by a geographical distribution of certain peers. This option can be achieved through an external service, such as MaxMind [42], which provides some information about the hostname, country code and name, region code, etc. The hashing algorithm should order peers in the overlay according to their country and region code (location ID). Therefore all user IDs automatically are generated with the location ID appended next to the domain name (ex. user_id@domain.location_id).

Because of the architectural preliminaries of the two tier overlay, mailing operations are divided accordingly: lower tier overlays for storing/replicating email content and super node overlay for notifications. For replicating email content across lower tier the solution adopted in [47] is required. Through this implementation every node evaluates itself according to the time spent to the mailing system. The time is measured in counts (2 minutes) over an interval of 24 hours and every node builds an availability chart according to its uptime evaluation. Every node from the lower overlay knows information about few others availability chart through synchronizing this information through the finger table, successor and predecessor. Because of the geographical distribution of peers within lower tier overlay, time zones discrepancies are not handled in the replicating process of email content.

In the super node overlay data availability is guaranteed from the Chord protocol. Replication of email notifications across this side of the overlay are not required because its member peers are present in the network with more than average resources, such as computing power, uptime, shared space and bandwidth speed.

The email notifications are stored separately for each user id in an inbox holder. This holder is represented through a list of new and old entries of email notifications and the public PGP key used for encrypting email content when emails are to be sent to this address (user id). Each entry consists of the sender address, subject message, and addresses of nodes within lower tier overlay where the email is replicated according to a generated availability chart. The authors [64] considered that every node stores one at the time notifications in the inbox holder without being susceptible to concurrent writing.

6.3.2. Email Operations

In Figure 6.9 Alice's computer represents one of the member peers of the mailing architecture design situated in one of the lower Chord tier. Alice uses her common email client to write an email to Bob (*sending operation*). When the email is sent from the MUA side, the P2P mailing application receives the email according to a RFC standard format and converts it to an internal one.

Alice's mailing application passes the request to the lower tier overlay for backing up the email content. First the public PGP key is fetched from the inbox holder named after Bob's email ID. For this purpose, the node situated in the lower overlay, represented by Alice's computing machine, searches in the super node overlay for Bob's inbox holder and retrieves the public PGP key (Figure 6.9 strong line arrows). The email content is encrypted according to the fetched key and the replication process begins in the same lower tier identified by the Alice's geographical location ID.

The dotted lines in Figure 6.9 represent the target nodes used for replicating the email content. The replication process takes place through building another availability chart for the email content requested for sending. The process finishes when all gaps of 24 hours are filled with the corresponding uptime of targeted nodes used for replication.

The email content is stored under the successor of the node that performs the request (Alice's computing machine) and other nodes from the finger table. Every replication process occurs once at the time. Every node contacted as backup target checks also its neighbour nodes to fill up the gaps from the availability chart.

If other possible backup nodes are found, the targeted node responds to the requester with the corresponding address.

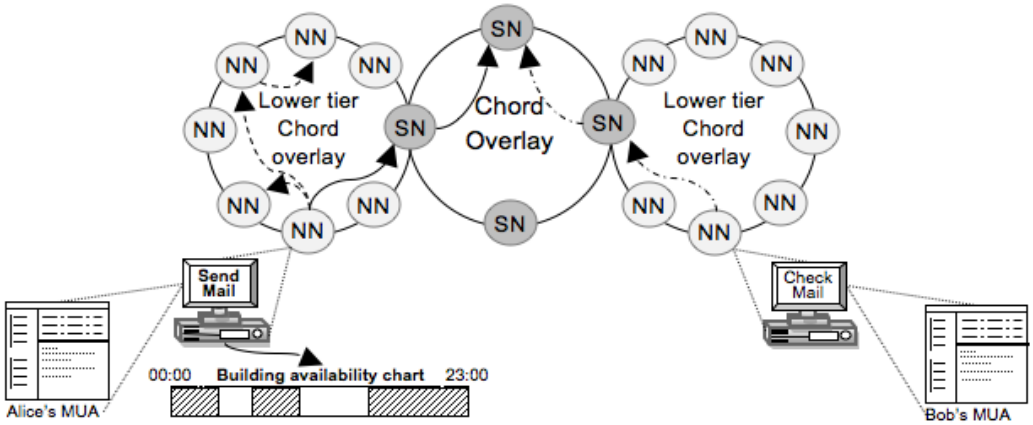


Figure 6.9 DMail Architecture Design

If none of the lower tier nodes are available for replicating Alice's email content (overwhelmed in terms of processing stress, uptime availability, etc.), the email request is passed to one of the super nodes from the upper tier. The super node becomes the new email sender requester and the same replication process takes place within its lower tier overlay.

After the completion of the replication process, the requester node fetches again Bob's inbox holder and stores the email notification with the corresponding information: from, to, subject and replication nodes addresses.

Retrieving the new email for Bob's user begins by connecting locally to the mailing daemon from the MUA application. Once the connection with the other peers was established, Bob's computing machine fetches his inbox from the super node overlay and Bob receives the new email notification (Figure 6.9 dashed and dotted arrows). The content will be downloaded locally from the nodes which addresses are appended in the notification. The download process will use all of the nodes available on the network. After the downloading process has successfully completed, Bob uses his private PGP key, which is securely stored on his computing machine, for decrypting the email content.

The *deletion operation* is implemented on every node from the overlay. Any data item that will be stored is marked with a number of counts measured in days. Every day passes by, the count is decremented, and when it reaches zero the stored data will be automatically deleted. The email content is referred here as every data item, notifications entries and inbox holders. If one user does not check his inbox for a period of time, all data will be automatically deleted from the overlay network.

The user can explicitly delete the email content from the replica nodes trough appending the request to the garbage collector holder placed on the super node overlay. When nodes join the overlay, they first check the garbage collector holder to see if certain data requested for deletion is available. If matches were found, the data is removed from the nodes and also the garbage collector is updated accordingly.

6.3.3. Interoperability and Applicability Solutions

The authors from [64] have considered two scenarios in handling operations with the traditional mailing service: standalone and hybrid mailing service.

For the considered scenario of standalone P2P mailing application, a small number of super nodes are required, registered as MX-Hosts in the domain name system (DNS) [52] named after the architecture solution: DMail.com. Trough this approach this solution design must handle only incoming mails from the traditional architecture. This requires that the nodes registered as MX-Hosts should bind their SMTP connector to an IP associated with the DMail domain. When an email is sent to this architecture from the traditional service, the DMail domain system is queried for retrieving the MX-hosts and emails are sent via SMTP protocol to the registered super node. The super node then performs the sending operation internally, described earlier in this paper.

The hybrid model of the DMail mailing architecture design is built through the collaboration between the P2P and Client/Server structure model. The DMail domain includes MX-entries for both the adopted models: datacenter gateways for the traditional one and super node hosts for the P2P architecture. When an email is sent to this architecture design, the designated MTA retrieves the MX-hosts for this purpose from the DNS. It receives a lists of hosts prioritized according to the P2P-traditional order. If none of the super nodes are reachable, overwhelmed in terms of storage space, bandwidth usage, computational power, etc., the MTA choses the hosts from the traditional model. This way, a focus is set more on the P2P model where the costs needed for handling the mailing operations are reduced than on using the traditional one, in terms of computing resources, specialized personnel and dedicated buildings. When an email is to be retrieved from this architecture design a mail relay should be designated for this purpose. The mail relay can point to both P2P and traditional data holders according to the used location for storing the email content.

As the society heads towards mobile technology and the Internet becomes more as a given resource, users will easily be able to use more efficiently P2P technologies without any restrictions of mobility issues, network availability and low uptime expectation. The DMail architecture concept can be applied without constrains on the available devices today. One existing user from this mailing service must explicitly select a target device for processing email operations such as: mobile phone, desktop computer, tablet, etc. With this set, the user can also access the mailing service from another device without the need of sharing resources or syncing email content again. Suppose Bob had set his desktop computer as the target device for contributing to the P2P mailing service. Bob can sign in through his mobile device to the mailing service, by specifying the targeted device address, without the need of sharing resources over the Internet again. When emails are sent to Bob’s inbox address, Bob can be notified from two sources: super node overlay and target device. This concept reduces the issues raised when no replica node of email content is available on the network.

6.3.4. Simulation and Experimental Results

The authors [64] simulated the mailing architecture design in an object oriented environment, where peers have been represented as objects within the application. Because peers can join or leave the network at anytime, case scenarios

were considered for each of the uptime probabilities: 0.1 where the entire network simulation depends on peers that join the network only for a short period of time to check their mail inbox status (5 to 10 minutes uptime); and 0.9 where peers remain connected to the network for several hours daily (highest uptime). The results were obtained through simulating a network environment of 10000 nodes.

Previous research authors have analysed their implementations [55][56] according to the number of email copies and download speed. The authors used in their work a new metric that they recently introduced in a previous research [61], which specifies the availability of a replicated email.

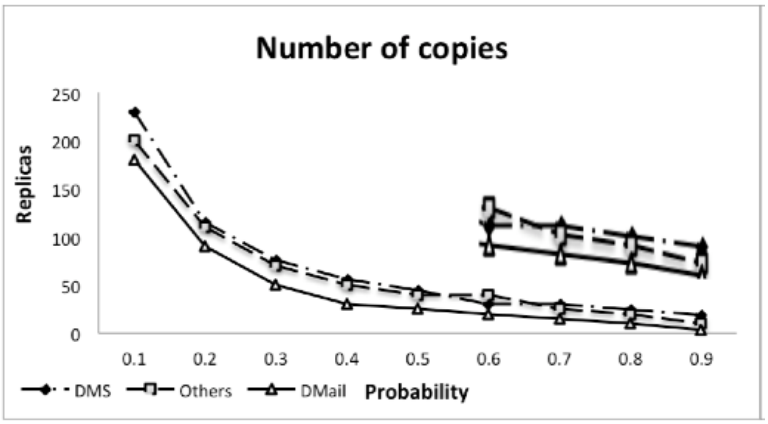


Figure 6.10 Number of Email Replicas

In Figure 6.10 the authors analysed how many replica nodes are needed to assure content availability for a sent email. The more nodes are involved in the replication process, the overall mailing service performance decreases as well, as a result of network traffic, storage stress and computing power usage. The DMail architecture design requires a smaller number of email replicas and therefore performs better due to the adopted overlay model and method of uptime prediction: every new email is replicated according to an availability chart of 24 hours interval located in the same geographical area, where no time zone discrepancies are implied. A significant improvement was achieved regarding the number of copies in terms of 22% less nodes involved in email content replication compared to Secure Email[55], Experimental Mail [56] and DMS [61].

In Figure 6.11 the email content availability was analysed over an interval of 24 hours per day. Through this metric the optimum number of email copies presented in Figure 6.10 were specified. The results show significant improvement compared to [61]. A overall performance evaluation of 17% was gained, better availability of email content. When relying on users that sign in to the mailing service only for a short period of time (0.1 uptime probability), a benefit up to 6 hours per day of email availability was achieved. Through raising the uptime expectation of joining peers, the email availability increases as well, tending to a constant value that reaches 24 hours of email availability daily.

The authors shown in [61] that the download speed is not always proportional with the number of peers selected for download and it is also limited according to the Internet service provider specifications (limited network traffic). Therefore the authors believe that download speed cannot be considered a metric

according to base our research results upon. In practice, uptime probabilities ranging from 0.5 to 0.7 are to be considered mostly, covering all the discussed user types.

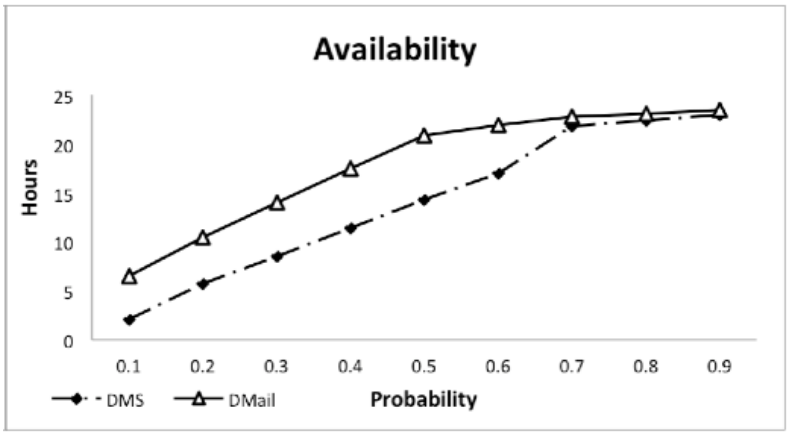


Figure 6.11 Average Email Availability

In the work of [64] the authors designed a new mailing architecture solution based on the P2P network model. They provided two types of interoperability with the traditional mailing service and an integration solution with the up to date technology. The simulation results show that the mailing architecture design can handle well any type of requests from other Client/Server mailing services and also provides a stable data holder for each of the considered uptime case scenarios.

7. Conclusions

This chapter focuses on highlighting the original contributions added throughout this thesis and also on concluding the experimental results obtained for this purpose. Through highlighting the thesis contributions I will refer also to the decisions taken towards designing one *distributed mailing system*. Although several mailing architectures were described in this thesis, the goals mentioned in the first chapter were met on every proposed implementation. The experimental results highlight mainly the strengths of every developed mailing concept. Our main scope is to prove that such mailing system architectures can be easily adopted and implemented on today's devices.

7.1. Original Contributions

I have chosen the name *Distributed Mailing System* for describing the thesis theme in the simplest way, so it can be easily understood and the concept I have developed throughout the 3 years of research activity can be anticipated. Yet one simple designation reveals several research aspects for implementing such a complex architecture design.

The Peer-to-Peer network concept sets the basis for our research. The second chapter of this thesis provides a brief presentation of the network principles. Through this chapter, elementary information regarding the used protocols in handling the mailing architecture operations was presented. I cannot claim authorship for most of the presented information, but an original contribution is represented through my vision regarding the Peer-to-Peer network concept present in the last part of chapter two.

Chapter three provides a state of the art introduction into the Peer-to-Peer implementations available today. Through presenting and analysing several overlay P2P concepts, I have highlighted my contribution by implementing a new approach in handling peers over the network. My model proposal is built as a framework support for several P2P applications, facilitating the operations of inter-communication, resource evaluation and customization of the overlay according to the application predominant characteristics. By this approach several applications can be built across the same framework support, every implementation being able to configure several virtual overlays across the one who serves as a common network support.

In chapter three I also provided an analysis regarding the various uptimes characteristics of peers throughout different applications types. I cannot claim authorship regarding the measurements taken for each of the analysed implementation, but throughout the investigative part I developed an algorithm capable of predicting the moment in time when a certain peer will join or leave the P2P network. The proposal is futile in handling caching techniques across the unstable environment of P2P network, reducing dramatically the number of replica nodes used in handling data availability and consistency across such network types.

Chapter four handles the current mailing architectures available today. A state of the art introduction regarding the mailing implementations based on both the server-centric and Peer-to-Peer architecture was presented. My contribution to this chapter resumed in providing a top view regarding the available architecture models available today. Through this investigation I was able to develop new

mailing architecture designs, improving the mechanisms of the current P2P implementations.

Most of the current Peer-to-Peer mailing architectures were designed to be compliant to a certain RFC standard format. Although their architecture implementation relies on a P2P model, the operations of handling email content are conform to the traditional protocols provided in chapter two. Chapter five handles the issues raised by such implementations through an interface model proposal that enables and encourages future mailing implementations developed across the P2P network to handle all internal operations separately from the traditional architectures. By this approach P2P mailing architectures have to handle only outgoing or incoming email operations from or to the traditional mailing systems available today. My contribution to chapter five yields through designing such an interface model, in this manner the future implementations of P2P mailing implementations will also rely on a self-developed protocol format without the concern of inter-compatibility with the traditional mailing concept based on a server-centric model.

The research activities performed in chapter six are entirely original. In this chapter I have proposed three mailing architecture types that perform better than the ones analysed in chapter 4. For this matter, I used the previous results provided by chapters three and five for designing the mailing architectures. All the new approaches in handling email operations over the P2P network were implemented according to the uptime prediction algorithm provided in chapter three and interoperability interface provided in chapter five.

A special contribution yields from the research activity carried on during the three years of developing and improving Peer-to-Peer mailing architecture. A major effort was made in developing, designing and researching this domain of mailing architecture developed across the unstable environment of Peer-to-Peer networks.

7.2. Analysis of the results

In this subchapter I will focus mainly on the results obtained after developing new architectures (P2P overlay frameworks and mailing systems) and argue our contributions.

In chapter three I proposed an extension [41] of the Chord overlay [19] that enables applications to configure their infrastructure according to the computing resources available throughout the Peer-to-Peer network. Chord, like most of the available overlay models, is built in the manner of handling every node as equal in the network. According to the analysis provided also in chapter three, I concluded that the implementations that harness the heterogeneity of nodes within the P2P network perform better in terms of search queries, data availability and consistency, stable network backbone, etc. Most of the analysed implementations that perform operations under such circumstances, promoted nodes with above average computing resources (increased uptime, computing power, bandwidth, etc.) as super nodes. Any other node that was not eligible for this category was threated as an ordinary node. Trough the proposal present in chapter three, I wanted to design an overlay that facilitates applications to build virtual layers gradually, according to the resource selection of nodes within the network. The hierarchical layers built on these requirements are handled into restrained entities over Chord, called hierarchical modules (HMs). Every HM is configured and identified by a control file

stored on the original Chord overlay (Base Chord Overlay). Trough this method of harnessing the most of the available computing resources throughout the P2P network presents also some benefits such as task balance between the layers, possibility of securing a certain layer level, data caching techniques are implemented on carefully selected nodes for this purpose, etc.

The hierarchical extension of Chord [41] was simulated on the Oversim [43] platform. The framework provided by the Oversim platform facilitates the environment of building and testing P2P overlay models. The proposed hierarchical extension has performed well on the simulation environment. The measurement of lookup operations represents one of the most critical evaluation criteria in evaluating an overlay model. The proposed model requires additional lookup operations resulting in finding first the associated control file of a certain HM and then the lookup operation can be performed on the queried module. Although this hierarchical model requires additional lookup operations, the efforts are minimal when compared to other implementations that provide extensions for the same Chord overlay.

The interface design that handles protocol interoperability between Client/Server and Peer-to-Peer mailing systems is strictly theoretical. No simulation was required for proving the reliability of such an interface design. The scope of such a proposal was to encourage future P2P mailing architecture to develop the mailing mechanisms compatible only when handling operations outside the P2P network.

In chapter six, three mailing architecture types were proposed, every implantation having an unique approach in handling mailing operations across different P2P architectural types: DMS [61], HMail [63] and DMail [64]. As a common goal for building the mailing systems, I was focused first on achieving a stable environment across which mail operations can be performed safely on the P2P network environment.

DMS. The architecture design across which I developed the DMS architecture is based on the unstructured overlay model present in [62]. The mailing architecture present in [61] is structured according to the community validation. Every community is formed through connecting super nodes within several ring topologies. The email operations are handled onto a three tier overlay network: ordinary nodes, ordinary community and dispatch community. The communities are handled in restrained areas according to the GeoIp tagging of every participant to the network. The mailing operations occur between communities as follows: caching techniques occur only on the ordinary nodes and notifications are stored on the community layer. Because of the unstructured overlay model adopted in designing the DMS architecture, a third tier is used for handling operations throughout communities within different geographical locations.

The simulation environment across which I tested the DMS implementation is based on an object-oriented framework. The participants were simulated according to various uptime case scenarios: 0.1 represents the worst case scenario (users join the network only to check the inbox status) and the best case scenario is represented by 0.9 describing the users who remain for several hours logged to the P2P network. The obtained results show that although caching techniques perform best by distributing (dividing) the tasks across several tiers, the architecture design lacks in terms of limitation of the queries across the unstructured P2P network. Because every query sent outside the communities handled into the same geographical distribution is limited according to a TTL (time to leave) flag, the bandwidth latency can be overloaded by broadcasting such message types.

Nevertheless, the mailing process is well balanced among users and the operations of sending emails are not represented by persistent tasks. Hence I conclude that this mailing system performs well in any of the situations generated by the P2P network implementation.

HMail. The mailing system presented in [63] was developed across the overlay model [41] described in chapter three. The hierarchical overlay model extends Chord [19] by building several virtual layers across the platform already established (Base Chord Overlay). Every virtual layer is built through connecting several nodes from the layer underneath according to a validation process of certain property (computing resources evaluation, etc.). By this approach, several restrained entities are developed across the original Chord overlay, with a predefined number of hierarchical layers called hierarchical modules (HM). Every HM is configured through a control file stored on the lowest layer (original Chord overlay) and the lifetime of such entity is limited, according to the valid entries of certain nodes validated by the requirements established by the control file.

HMail defines several hierarchical modules validated by the geographical distribution of joining peers. Every HM has two layer levels, the first one being validated by nodes with above average resources of bandwidth and uptime; and the second one is formed through connecting nodes within the first layer with increased CPU power and shared space. The first layer of a certain hierarchical module is used as a *mail submission agent* and *mail retrieval agent*. This layer interacts directly with the user handling the operations of sending and retrieving the email content. The second layer is divided in three overlays: spool area, mailbox area and activity monitor. This layer handles incoming and outgoing email content providing all the operations of *mail transfer agent*, garbage collector and public PGP keys used for encrypting the email content. The sending process occurs safely between HMs according to the validation of random data encrypted/decrypted according to the PGP keys.

The solution presented in [63] represents one of the most complex mailing architectures design, where every process involved in the traditional solution can be found also across the P2P mail implementation. Throughout developing this new mailing architecture, I considered also additional security facilities in handling the mailing operations. By this approach, a limitation is set on every hierarchical module which provides restriction mechanisms for any unauthorized access for unregistered participants to the mailing system. The simulation results show significant improvements than DMS where the email content is replicated across the nodes with above average computing resources: shared space, bandwidth, uptime and CPU power. The address space has no limitations in handling queries from every point of the network and further, if a certain HM cannot be sustained from its nodes anymore, the email content is distributed automatically across the lowest Chord layer (Base Chord Overlay). The drawbacks of this implementation is triggered by the high resource expectations of a certain HM. By this, it becomes difficult for a node to re-join a certain hierarchical layer. This is possible due to joining time offsets of certain peer in the network, which sometimes yields with a lower resource evaluation than expected.

DMail. The mailing system presented in [64] is developed across the two tier overlay available in [36]. The two tier overlay model handles nodes with above average resources in a distinct Chord overlay from the others formed through connecting ordinary nodes. The tasks used in handling the email operations are evenly distributed among tiers: the email content is safely stored on the lower tier overlays and notifications are kept on the tier sustained from super nodes. The store

process involves replication of the email content on several ordinary nodes according to the algorithm presented in the third chapter of this thesis. The sending and receiving processes occur by placing or retrieving notifications from the main overlay tier (SN overlay).

Trough handling all the notifications centralized and separately from the lower tiers this implementation design is similar with the traditional mailing service available today. By handling the mailing architecture in this manner, I was able to extend the features of this concept by proposing an applicability model for today's mobile technology evolution. As the society heads towards mobile technology and the Internet becomes more as a given resource, users will easily be able to use more efficiently P2P technologies without any restrictions of mobility issues, network availability and low uptime expectation. The DMail architecture concept can be applied without constrains on the available devices today. One existing user from this mailing service must explicitly select a target device for processing email operations such as: mobile phone, desktop computer, tablet, etc. With this set, the user can also access the mailing service from another device without the need of sharing resources or syncing email content again.

The DMail architecture model was simulated in an object oriented environment that I personally developed for this matter. All the information considered in this model was handled as objects within the environment: participants to the network (peers), email content and notifications, metric characteristics and network topology. The obtained results show a significant improvement than the other proposed mail system models in terms of email availability and accessibility, stable network environment and overall increased performance in handling computing resources across the P2P network. This concept design represents one of the closest implementations to the traditional mailing service, where an attempt to centralize information within a decentralized network model was made possible by harnessing the benefits of the overlay model presented in [36].

7.3. Published Papers and Impact

My contribution in the domain of *Distributed Mailing System* is reflected in the mentioned articles:

- **P. E. Mezo**, M. Vladutiu and L. Prodan, "Design of a Hierarchical based DHT Overlay P2P routing Algorithm", 11th IEEE International Conference on Computer and Information Technology, Paphos, Cyprus, Aug. 2011, pp. 415 – 420, ISBN: 978-1-4577-0383-6 (BDI, IEEE rank).

- **P. E. Mezo**, M. Vladutiu and L. Prodan, "Interoperability solution between Peer-to-Peer and Client-Server based mailing systems", 2011 IEEE 17th International Symposium for Design and Technology in Electronic Packaging (SIITME), Timisoara, Romania, Oct. 2011, pp. 45 – 48, ISBN: 978-1-4577-1276-0. (BDI, IEEE rank).

- **P. E. Mezo**, M. Vladutiu and L. Prodan, "Distributed Mailing System (DMS)", 2011 IEEE 17th International Symposium for Design and Technology in

Electronic Packaging (SIITME), Timisoara, Romania, Oct. 2011, pp. 349 – 354, ISBN: 978-1-4577-1277-7. (BDI, IEEE rank).

- **P. E. Mezo**, M. Vladutiu and L. Prodan, "HMail: A hybrid mailing system based on the collaboration between traditional and Peer-to-Peer mailing architectures", 2012 IEEE 7th International Symposium on Applied Intelligence and Informatics (SACI), Timisoara, Romania, May. 2012, pp. 255 – 260, ISBN: 978-1-4673-1014-7. (BDI, IEEE, Australian Research Council list class C rank).

- **P. E. Mezo**, M. Vladutiu, L. Prodan and F. Opritoiu, "DMail: Distributed mailing system based on the collaboration between traditional and Peer-to-Peer mailing architectures", 2012 International Conference on Information Engineering, Lecture Notes In Information Technology, Vol. 25, Singapore, Singapore, Jun. 27-28, pp. 128 -135, ISBN: 978-1-61275-024-8. (Ei Compendex, Cambridge Scientific Abstracts, Google Scholar, IEE, ISI rank).

Two Ph.D. reports were presented in the Computer Science and Engineering Department, "Politehnica" University of Timisoara:

- **P. E. Mezo**, M. Vladutiu, L. Prodan, F. Opritoiu, "Distributed Mailing System", Ph.D. Report 1, "Politehnica" University of Timisoara, December 2011, pp. 1-60

- **P. E. Mezo**, M. Vladutiu, L. Prodan, F. Opritoiu, "Distributed Mailing System", Ph.D. Report 2, "Politehnica" University of Timisoara, July 2012, pp. 1-70.

7.4. Future Work and Research Direction

The subject of Distributed Mailing System leaves open several research fields for further debate. Although I have tried to include all the sub domains into the research, I could only reach a minor part of the whole mailing mechanism developed across the Peer-to-Peer environment.

One research direction would focus on generating a stable environment across the Peer-to-Peer network. This research field includes network topology design and simulation, uptime status behaviour analysis of the joining peers and scalability of such model design. These three fields are close related because this research direction is co-dependent on every one of them. The network topology should consider the peer uptime behaviour for building distinct tiers in handling the mailing operations, separately from the tier that provides a common access to all the participants. Also the need of scaling a new concept of network topology remains one of the major challenges in handling such architecture types. I also encourage future extensions of my work to provide a common framework for several applications across the P2P network. In this manner, every application design can be inter-compatible with others, in terms of computing resources across the network, implemented protocols, restricted storage area, etc.

Throughout developing the mailing system, I provided a minimal security solution to guarantee only the email content. Future research directions could

extend my model by addressing the security at communication level, or further, by using existing certificate authorities for this matter.

As our society moves towards new mobile technologies, the traditional mailing architecture will also suffer some changes. But as discussed in this thesis, the changes added along with the current technology were minor, addressing only the secure connection between entities. Another research direction that should extend my work can refer to the newest implementations of traditional mailing architectures. An interoperability solution can be developed for this matter, by providing a two-side compatibility with the traditional architectures: handling incoming and outgoing emails.

The newest trend in handling the mobile technology across the network is to handle all the data into the "Cloud". The computer society suggests that the current mobile technology lacks drastically in terms of computing power and this has generated a solution of handling all the information on a server-centric architecture. This solution not only demands high costs for implementation, but also generates high usage of bandwidth latencies. The Peer-to-Peer model remains a cost effective solution for this matter and by constantly improving its mechanisms, the bandwidth usage can be limited for such network architectures types.

I will continue this research activity in developing new mailing architecture models across the Peer-to-Peer network environment. I will focus also on implementing a version of the provided and discussed models across the current technologies available today.

References

[1] R. Dingledine, M. Freedman and D. Molnar, "PEER TO PEER: Harnessing the benefits of Disruptive Technologies", s.l. : O'Reilly Media, Feb 2001. ISBN:978-0-596-00110-0.

[2] K. G. Coffman and A. M. Odlyzko, "Growth of the Internet", AT&T Labs – Research, Preliminary version July 6, 2001.

[3] William Stallings, "Operating Systems Internals and Design Principles", Fifth Edition, © 2005 Prentice Hall of India, New Delhi – 110 001,2006, ISBN:81-203-2796-9.

[4] Stefan Saroiu, P. Krishna Gummadi and Steven D. Gribble, "Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts", Department of Computer Science and Engineering, University of Washington, Seatle, Mar. 31, Oct.

[5] Clip2 Distributed Search Services. *"The Gnutella protocol specifications v0.4".* 2000.

[6] M. Ripeanu. *Peer-to-Peer Architecture Case Study: Gnutella Network.* University of Chicago, Computer Science Department.

[7] I.Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system", Berkeley,CA , USA : In Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability, Jun. 2000.

[8] N. Leibowitz, M. Ripeanu, and A. Wierzbicki, "Deconstructing the Kazaa Network", SantaClara, CA : 3rd IEEE Workshop on Internet Applications (WIAPP'03), 2003.

[9] Yoram Kulbak, Danny Bicson and academic supervisor Prof. Scott Kirkpatrick, "The eMule Protocol Specification", January.

[10] D. A. Turner and K. W. Ross, "Continuous media e-mail on the internet: Infrastructure inadequacies and a sender-side solution", IEEE Network, 14(4): 30-37, July/Aug 2000.

[11] D. A. Turner and K. W. Ross, "A comprehensive architecture for continuous media email", IEEE Multimedia, 8(2): 88-98, Apr/June 2001.

[12] Ananth Grama, George Karypis, Vipin Kumar and Anshul Gupta. "Introduction to Parallel Computing", Second Edition. s.l. : Addison - Wesley, 2003. ISBN-10: 0201648652.

[13] C. Douligeris and P. Kotzanikolaou, "Network Security - Telecommunication Systems and Technologies", Vol. II, pp. 19, © Encyclopedia of Life Support Systems (EOLSS).

[14] Microsoft, "Implementing a Microsoft® Windows Server ™ 2003 Network Infrastructure: Network Hosts", © 2005 Microsoft Corporation.

[15] Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network", U.C.Berkeley, CA : Technical Report, TR-00-010 , 2000.

[16] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, descentralized object location, and routing for large-scale peer - to - peer systems", Heidelberg, Germany : Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Nov 2001.

[17] A. I. T. Rowstron and P. Druschel, "Storage management and caching in PAST, A large-scale, persistent peer-to-peer storage utility", Banff, Al- berta, Canada : Proceedings of the 18th ACM Sympo- sium on Operating Systems Principles (SOSP), Oct 2001.

[18] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-tolerant widearea location and routing", U.C.Berkeley, CA : Technical Report UCB/CSD-01-1141, 2001.

[19] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Bal- akrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications", s.l. : Technical Report TR-819, MIT., Mar. 2001.

[20] S. Androutsellis-Theotokis and D. Spinellis, "A Survey of Peer-to-Peer Content Distribution Technologies", Athens University of Echonomics and Business, ACM Computing Surveys, Vol. 36, No. 4, December 2004, pp. 335-371.

[21] S. A. Baset and H. G. Schultzrinne, "An analysis of the Skype Peer-to-Peer Internet Telephony Protocol, Department of Computer Science, Columbia University, New York NY 10027.

[22] W. Sulliwan, D. Werthimer, S. Bowyer, J. Cobb, D. Gedye and D. Anderson, "A new major SETI project based on Project SERENDIP data and 100,000 personal computers", Proceedings of the 5th International Conferemce on Bioastronomy, 1997.

[23] ***.GenomeAtHome 2003. http://genomeathome.stanford.edu.

[24] A. V. M. Keromytis and D. Rubenstein, "SOS: Secure overlay services", Proceedings of the ACM SIGCOMM'02 Conference, Pittsburgh, PA., 2002.

[25] P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini and I. Zaihrayeu, "Data management for Peer-to-Peer computing: A vision", Proceedings of the Workshop on the Web and Databases (WebDB'02).

[26] R. Huebsch, J. HellerStein, N. Lanham and B. Thau Loo, "Querying the Internet witj PIER", Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003.

[27] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, S. Gummandi, H. Weatherspoon, W. Weimer, C. Wells and B. Zhao, "Oceanstore: An architecture for global-scale persistent storage", Proceedings of ACM ASPLOS, 2000.

[28] D. Goldschlag, M. Reed and P. Syverson, "Onion routing for anonymous and private Internet connections", Comm. ACM 42, pp. 39-41, 1999.

[29] L. Xiong and L. Liu, "Building trust in decentralized peer-to-peer communities", Proceedings of the International Conference on Electronic Commerce Research, 2002.

[30] ***.SkypeOut. http://www.skype.com/products/skypeout

[31] ***.SkypeIn. http://www.skype.com/products/skypein

[32] ***. Global IP Sound. http://www.globalipsound.com

[33] J. Daemen, V. Rijmen, *The Design of Rijndael: AES – The Advanced Encryption Standard.* Springer, 2002.

[34] J. Rosenberg, J. Weinberger, C. Huitema and R. Mahy. *STUN: traversal of user datagram protocol (UDP) through network address translators (NATs).* RFC 3489, IETF, Mar. 2003.

[35] J. Rosenberg, C. Huitema and R. Mahy, "TURN: traversal using relay NAT", Internet draft, Internet Engineering Task Force, September 2005.

[36] M. Pandey,S. Mushtaq Ahmed, B. D. Chaudhary, "2T-DHT: A Two Tier DHT for Implementing Publish/Subscribe", s.l. : International Conference on Computational Science and Engineering, 2009.

[37] Prasanna Ganesan, Krishna Gummadi and Hector Garcia-Molina, "Canon in G Major: Designing DHTs with Hierarchical Structure," s.l. : Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04), 2004.

[38] Zhiyong Xu, Rui Min and Yiming Hu, "HIERAS: A DHT Based Hierarchical P2P Routing Algorithm", s.l. : Proceedings of the 2003 International Conference on Parallel Processing (ICPP'03), 2003.

[39] Giscard Wepiwe and Plamen L. Simeonov, "A Concentric Multi-ring Overlay for Highly Reliable P2P Networks", s.l. : Proceedings of the 2005 Fourth IEEE International Symposium on Network Computing and Applications (NCA''05), 2005.

[40] P. Zimmermann, "The Official PGP User's Guide", The MIT Press, 1995.

[41] P. E. Mezo, M. Vladutiu and L. Prodan, *Design of a Hierarchical based DHT Overlay P2P routing Algorithm*, 11th IEEE International Conference on Computer and Information Technology, pp. 415 – 420, Aug. 2011. Paphos. Cyprus.

[42] ***.MaxMind.http://www.maxmind.com.

[43] Ingmar Baumgart, Bernhard Heep and Stephan Krause, "Over{S}im: A Flexible Overlay Network Simulation Framework", Ak, USA : Proceedings of 10th IEEE Global Internet Symposium, May 2007.

[44] F. Standard, "1037c: Glossary of Telecommunications terms", The Institute for Telecommunications Sciences, Oct. 2006.

[45] J. Liang, R. Kumar and K. W. Ross, "Understanding Kazaa. Department of Computer and Information Science", Polytechnic University Brooklyn, NY 11201.

[46] S. Guha, N. Daswani and R. Jain, "An Experimental Study of the Skype Peer-to-Peer VOIP System", Cornell university and Google.inc.

[47] G. Song, S. Kim and D. Seo, "Replica placement Algorithm for Highly Available Peer-to-Peer Storage Systems", First international Conference on Advances in P2P Systems. IEEE 2009.

[48] B. Cohen, "Incentives build robustness in BitTorent. Workshop on Economics of Peer-to-Peer Systems", Vol 6, berkeley, CA, USA, 2003.

[49] ***.RFC Database.http://www.rfc-editor.org/rfc.html

[50] Sean Turner, Russ Housley, "Implementing Email Security and Tokens: Current Standards, Tools, and Practices", Indianapolis, IN 46256 : Wiley Publishing Inc. ISBN: 978-0-470-25463-9.

[51] J. Mcbee and D. Elfassy, "Mastering Microsoft Exchange Server 2010", Copyright © 2010 by Wiley Publishing, Inc., Indianapolis, Indiana, ISBN: 978-0-470-52171-7.

[52] P. V. Mockapetris, "RFC 1035: Domain Names – implementation and specification", Nov. 1987.

[53] J. Robert, S. Czerwinsky, A. D. Joseph, E. A. Brewer and J. Kubiatowicz, "NinjaMail: the Design of a High-Performance Clustered, Distributed E-mail System", Proceedings of the 2000 International Workshops on Parallel Processing (ICPP'00 – Workshops), IEEE.

[54] Y. Zhao, S. Zhou, and A. Zhou, "E-mail services on hybrid P2P networks", In Grid and Cooperative Computing Conference, 2004.

[55] J. Kangasharju, K. W. Ross and D. A. Turner, "Secure and Resilient Peer-to-Peer E-Mail: Design and Implementation", Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P'03), 2003.

[56] E. Kageyama, C. Maziero and A. Santin, "An experimental peer-to-peer e-mail system", 11th IEEE ICCS, 2008.

[57] S. Bercovici, Y. Frishman, I. Keidar and A. Tal, "Decentralized Electronic Mail", Proceedings of the 26th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'06), 2006.

[58] O. Holder, I. Ben - Shaul, and H. Gazit, "Dynamic layout of distributed applications in FarGo", Proceedings of the 1999 international Conference on Software Engeneering. IEEE Computer Society Press, 1999.

[59] P. E. Mezo, M. Vladutiu and L. Prodan, "Interoperability solution between Peer-to-Peer and Client-Server based mailing systems", 2011 IEEE 17th International Symposium for Design and Technology in Electronic Packaging (SIITME), Timisoara, Romania, Oct. 2011, pp. 45 – 48.

[60] ***. Google. http://www.google.com/about/datacenters/locations/index.htm

[61] P. E. Mezo, M. Vladutiu and L. Prodan, "Distributed Mailing System (DMS)", 2011 IEEE 17th International Symposium for Design and Technology in Electronic Packaging (SIITME), Timisoara, Romania, Oct. 2011, pp. 349 – 354.

[62] A. Moravek and I. Jelinek, "Using Centralized Element in P2P Network For Better Community Management", International Conference on Computer Systems and Technologies - CompSysTech'2004.

[63] P. E. Mezo, M. Vladutiu and L. Prodan, "HMail: A hybrid mailing system based on the collaboration between traditional and Peer-to-Peer mailing architectures", 2012 IEEE 7th International Symposium on Applied Intelligence and Informatics (SACI), Timisoara, Romania, May. 2012, pp. 255 – 260.

[64] P. E. Mezo, M. Vladutiu, L. Prodan and F. Opritoiu, "DMail: Distributed mailing system based on the collaboration between traditional and Peer-to-Peer mailing architectures", 2012 International Conference on Information