“Politehnica” University of Timisoara

Distributed Mailing System
PhD Report I

Patrik Emanuel Mezo	*PhD Student*
Prof. Dr. Ing. Mircea Vladutiu	*PhD Coordinator*

ABSTRACT

This PhD Report describes the research activity carried on as part of the doctoral program entitled “Distributed Mailing System”. It describes the research activity carried on as part of the second year of the PhD program. Peer-to-Peer network infrastructure lies between the communication layer (network protocols) and the complex architectures that combine several standards in achieving structural stability in such unpredictable environment. This Report work presents the research directions of handling email architectures over the Peer-to-Peer network infrastructure. Traditional email architectures rely on server-centric design, having dedicated buildings for storage and trained personal for managing such complex system designs, resulting in high service costs. We intend to build an alternative for this architecture implementation, which relies on personal computing resources such as: bandwidth, computing pow er, storage space, etc. This enables each peer to effectively contribute to the mailing system according to the real evaluation of its resources, therefore increasing overall application performance and reliability.

TABLE OF CONTENTS

TABLE OF FIGURES

1. Introduction

Originally the internet was shaped in a Peer-to-Peer (P2P) manner, where every participant to the network was treated as equally, not in a master/slave or client/server relationship [1]. The first breakthrough that made internet possible was the ARPANET design in the late 1960's. The goal of this network was to share computing resources over the U.S. The challenge was to integrate different kinds of network to a single one network architecture.

Now, the internet is a shared resource, a cooperative network built out of millions of hosts all over the world. There are millions of applications that want to use the network, placing strain on the most basic of resources: bandwidth [2].

An important element of our society is communication. Since the beginning of men kind, 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 even gestures.

Nowadays one of the most common communication tools is electronic mail (e-mail or email). Built on a server – centric architecture [3], 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. Managing modern mailing architectures implies also an increased cost in terms of: dedicated buildings distributed geographically and specialized trained personnel for maintenance and quality of service.

Considering the overview presentation, we propose new architectures designs, where every personal computing resource contributes to a single application system and becomes a part of it. We entitled our work: "Distributed Mailing System (DMS)", through which a complex mailing architecture design was shaped by combining both the peer behavior, in terms of time spend over the internet, and computing resources evaluation methods. The distributed mailing system has no central unit managing connections and email content. There is no central server designed to serve a certain task. All peers (computing systems) fulfill 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 computer resource to a global community. We want to adopt 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 first 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, our goal is to build the proposed system from cluster to network architecture layer for an unstable distributed network environment, as outlined in this report.

The second direction of research is to design a protocol below the one used for e-mail communication layer, for load – balancing network communication between distributed network parties.

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

2. Peer-to-Peer

The P2P 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: VOIP, mailing systems, social applications, etc.

A number of implementation attempts were pursued to harness the resources of peers in several architecture designs among which we bring to attention the following: Gnutella [4][5], Freenet [6], Kazaa [7][8], and eMule [9]. With the demand of scaling at a large number of peers, the above mentioned architectures presented some structure flaws: one lookup query was limited by a time-to-live descriptor (Gnutella, Kazaa) 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. We mention here CAN [10], Pastry [11] [12], Tapestry [13] and Chord [14]. 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.

2.1 Peer-to-Peer Framework

One of the main issues raised during P2P applications development was concluded in terms of network accessibility, each peer being able to communicate within a limited TTL (time-to-leave) search area. Two approaches have solved this issue: hybrid P2P designs and overlay network layers. The first solution provides server-centric elements for managing the coordination of certain peers within the network, while the second one handles the peer community into a single identifier space, through which peers are able to perform queries correlated to keys from the same space. Because hybrid P2P networks do not specify a standard according to which peer should connect and communicate, and every implementation has its own structural performance contribution, we will discuss only the overlay framework platforms.

2.1.1 Peer-to-Peer Overlay Framework

The overlay architecture concept was designed as a framework for applications situated above the overlay layer, such as file sharing or VoIP applications (Figure 1). An application on top of the overlay framework 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, etc.

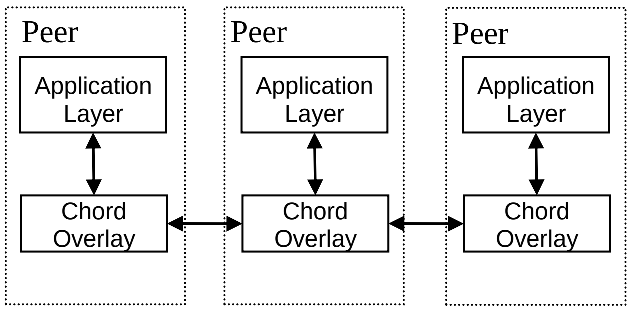

Figure 1 Overlay transparency for other applications

Chord is a structured P2P overlay network built on top of a ring model. Both nodes and keys are mapped under the same identifier space using a consistent hashing method [15]. The ring model can be viewed as a modulo 2^m identifier space, where joining nodes are ordered from 0 to 2^m-1. The available data stored under the Chord overlay is represented through hashing data IDs and obtaining a unique key ID in the identifier space, placed at the node with the same ID (or under its successor, if the node isn't present). The value of m should be chosen to fit a large amount of joining nodes in the identifier circle and thus preventing that two IP's or keys from the identifier space have the same hash ID. Each node from Chord is linked to its successor and maintains a list of size r of nodes following it in the ring. To accelerate the routing process, nodes in Chord point to a list of at most m successor nodes called a finger table. If node n points to the i^{th} entry from the finger table, then this entry designates a node located 2^i succeeding nodes away from the current position, where $1 \leq i \leq m$.

When a new node joins the network it asks an existing node from the Chord network to find an entry point in the ring. The existing node hashes the new node's IP and through the remote call of *new_node.init_finger_table(existing_node)*, it provides the new node with the information needed for joining the Chord network. At this moment the other nodes joining the network must be aware of the newly joined node by updating their finger table through *fix_fingers()* procedure. The *stabilization()* procedure ensures that a predecessor link is set to the newly joined node and new predecessor link is set for its successor to notice the presence of the new node in the Chord network. The periodical call of the *stabilization()* procedure ensures scaling Chord under churn through maintaining a valid finger list for each node participating to this overlay network.

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.

The Crescendo [16] 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 [17] 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 [18] 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 [19] 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.

2.2 Peer-to-Peer Mailing Architectures

Initially designed for academic purposes, the email application has become one of the most used tools that modern society has adopted at daily basis. Due to its original purpose, the email architectural structure has encountered several changes over time. Current mailing systems are built over a server-centric network architecture. The common model adopted for implementing email operations is store and forward. The mail user agent (MUA) represents the interface between the user and the mailing system. The transparency between the MUA and the mechanism behind the mailing process is gained through the coordination of several mail transfer agents (MTA). Through their cooperation, the MTAs assure several processes such as: receiving, storing, replicating and mail delivery to the local MUA via MTA. Hence, the whole mechanism is built according to a store and forward model, through which mails are forwarded until they reach destination (MUA or MTA).

Modern mailing systems architecture employs MTA at cluster level, where performance is gained by accessing and managing the whole architecture through distributing tasks among cluster resources. This approach has solved several design issues such as: data replication, location services, network availability or load balancing tasks, but with an increased cost. There are also scenarios that overcome the actual mailing architecture design [2][20] including: accessibility issues when a cluster lies behind an access link that is severed or flooded, storage stress due to multiple attachments and server processing stress.

The existing P2P mailing architectures were developed to overcome the issues that current mail architectures have to deal with. The main idea was to shape a system where the entire architecture depended on personal computing resources in a decentralized manner. Some of the implemented solutions were designed over the framework provided by overlay networks [10-14], through which mailing systems were developed to rely on homogenous computing resources from the identifier space. Other implementations were developed on hybrid architectures, combining the property of decentralized architectures with the server-centric design.

Current P2P mailing solutions were developed in both hybrid and overlay concepts. The first solution was able to coordinate peers through the presence of server-centric entities, and the second one mapped the entire community into a single identifier space, through which peers assigned each query to a key correlated with the same identifier space. All the solutions were designed to maintain the same functionalities as the traditional server-centric architecture.

The solution presented in [3] 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 approach used in [21] represents one of the best solutions concerning the P2P mailing architectures. The proposal was developed over the Chord overlay [5] 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 mailing architecture design in [22] 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 then being forwarded to the other nodes linked to the MTA node.

The solution presented in [23] 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.

3. Distributed Mailing System (DMS)

Our mail architecture model uses the concepts found in [21] [22] 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. Our proposal does not replicate 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.

3.1 Preliminary Assumptions

3.1.1 Architectural Preliminaries

The architectural model used for designing our mailing system relies on the concept used in [24], describing an interconnected multi-ring topology (Figure 2). 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 such as MaxMind [25]. Through the GeoIp tagging, we can clearly distinguish 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 [25]. 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) [26]. 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 our 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. The local

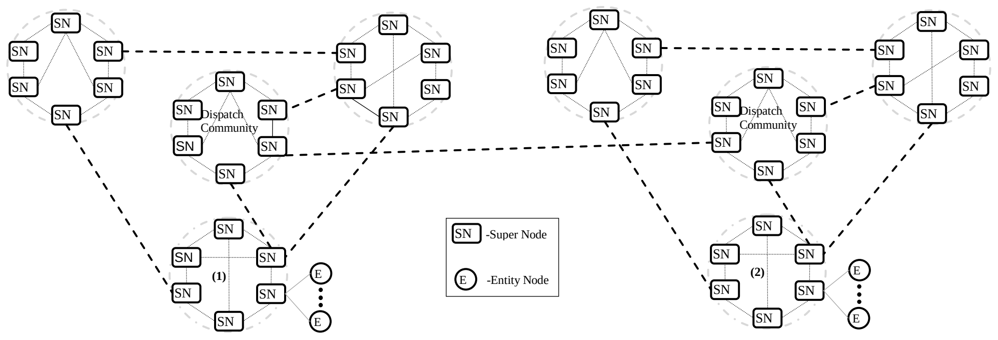

Figure 2 DMS Architecture Overview

connections are held between entity nodes and super nodes for assuring load balance among email operations and maintaining the mail service alive.

In Figure 2 we can clearly distinguish between the links used for interconnecting mail system parties to the network service. 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. Among the information provided by MaxMind, we can clearly distinguish the membership of one node to one certain country and the region inside that country. Because limited TTL broadcast messages are used in our mailing network architecture, we limited 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.

3.1.2 Metrics Preliminaries

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 [27]. Under these circumstances, in our architecture design we evaluate the bandwidth measurement 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 configuration. 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.

Uptime represents the key factor in data caching and replication. To provide a solid foundation for grouping participants, we designed the mailing architecture according to an uptime availability prediction. The concept found in [28] 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 neighbors 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 [28] could generate a good prediction within an interval of a week.

We designed our uptime availability as an average mean of a time slot of 60 minutes generating 24 time slots per day. Our 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, we are only interested in 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 3 represents one example of our method of analyzing 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 contribution of the peer to the mailing system.

Also the work present in [28] 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,

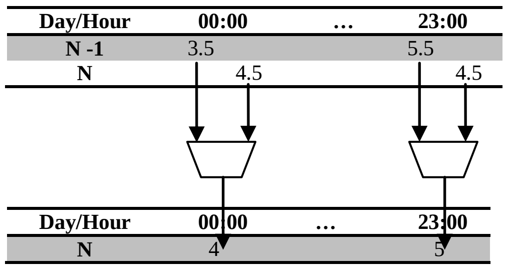

Figure 3 Uptime Prediction Evaluation

acording to MaxMind [25], 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:

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

3.1.3 Service Primitives

Throughout the evolution of traditional mail protocol, the RFC standard format [29] 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, our 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 [23], 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), we have to define the service primitives that help building the tasks: store, delete, fetch, append, read inbox and garbage collection. Throughout the mentioned primitives, we highlight also unmentioned key elements that shape our architectural design. For security purposes, we require the external services of PGP keys [30] implementation, assuring data security and user privacy. We also require that all user IDs 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).

3.1.4 Store Primitive

Due to our three layered architecture design, we define a store primitive 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 section III. Because the final computed score point highlights only peers with super node property among communities, a thorough evaluation is performed to fulfill 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. 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 section III. 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.

3.1.5 Delete and Garbage Collection Primitives

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 signaling 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 has not 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.

3.1.6 Fetch and Append Primitives

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.

3.1.7 Read Inbox Primitive

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. We are using 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, we require only to update a small amount of data for better quality services.

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. We implemented our mail service as being one of the store and forward type. Therefore, the user's MUA is in charge of replicating downloaded email content on the computing machine that served as a peer to our architecture design.

3.2 P2P Email Mechanism

For sending and receiving email content through our network architecture design, we appeal to the primitives defined in Section IV. We exemplify the email operations through the architecture overview present in Figure 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.

4. Interoperability solution between Peer-to-Peer and Client-Server based mailing systems

Analyzing the previous work on P2P mailing concepts, we have identified the need of implementing 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 contributes to the mailing system is considered at an abstract level for providing a reliable foundation for the P2P application concept. We encourage this way that 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 manner.

The mailing system represents a complex infrastructure of a series of precisely aimed tasks. Figure 4 shows a possible scenario for interconnectivity and communication within a mailing system. For an ease of understanding we have represented all the components that help clarify the mailing tasks outside the internet cloud. Naturally, one email provider has its data centers distributed according to geographical distribution (e.g. google.com [31]) to assure certain agreements, such as: service uptime, store and data availability, service performance, etc. In this example we use elements from different internet service providers (ISPs) 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 [25] 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 behavior of its peer members (uptime is unpredictable) it is very hard to determine a fixed address of such

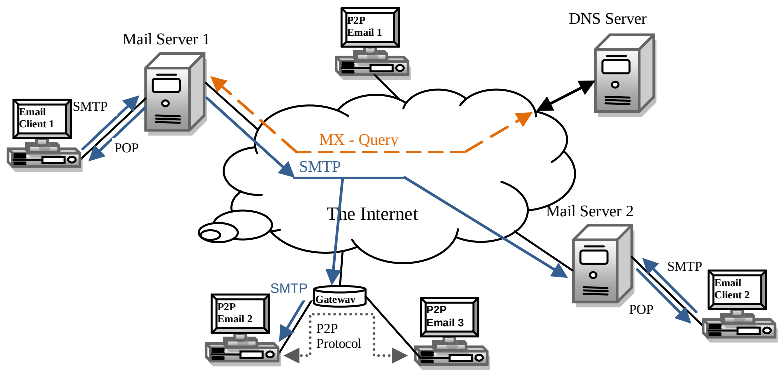

Figure 4 Interoperability Scenario between P2P and Client – Server based Mailing Systems

entities. Therefore, we present the situation in which the P2P mailing concept is present in an institution [23] which lies 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.

According to the case study presented in this section, we will provide an interface through which inter-compatibility with the traditional mailing systems is gained, and further, it also provides a second functionality with the commonly used mail client applications.

4.1 Architecture Implementation

The interoperability solution between P2P and server-centric mailing systems relies on the interface presented in Figure 5. 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. Because we encourage that P2P email content should travel according to a self-developed protocol, for separating the RFC standard from the internal P2P communication, we also provide a Peer Connector for that purpose. We provide

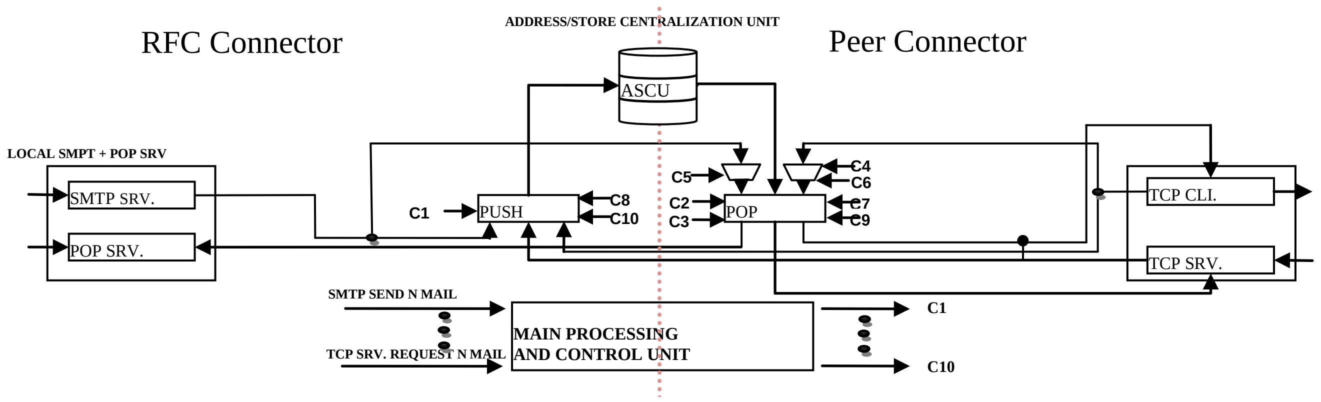

Figure 5 Interoperabiltiy interface between P2P and Client-Server based Mailing Systems

only solutions for the TCP/IP network layer; however, for any other protocol implementations, which are positioned higher or lower than the one presented in this work, the main process remains the same.

For handling the disk space every peer is willing to share, we provided the necessary connections to all the elements that help handle our 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 we implemented the interface as an additional application which serves as a service for the users of P2P mailing infrastructure, we have implemented the interface 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 2.

4.1.1 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 we have handled only the case of store and forward mailing system property, 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.

4.1.2 Peer Connector

There are several platforms through which the P2P mailing systems have been developed and improved. We have considered the Peer Connector 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. We have considered the TCP/IP network layer 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. We considered also the situation when one peer lies 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.

4.1.3 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) [14] method, data consistency should also be consider as a reliable way of handling data integrity. We have handled the ASCU entity as abstract in this paper work, because every P2P mailing implementation comes with a self-developed protocol through which data is also being handled according to a different format.

5. Conclusions

5.1 Experimental Results

We have implemented our architecture design in an object oriented environment, where both the entity and super node were handled as objects. We tried to bring our simulation closest to the research provided in [28], through which the users are being characterized though their behavior in time spent over the internet. Hence, we analyzed the possibilities ranging from the user who only remains logged in to the mailing service until it finishes reading emails (0.1 probability), to the user with an increased spent uptime (0.9 probability).

Figure 6 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. We are aiming in obtaining solutions precisely for situations in which all the participants act according to the uptime probability described in Figure 6. 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.

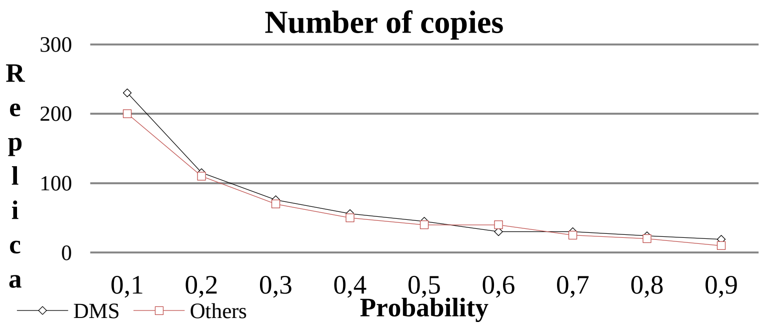

Figure 6 Number of email replicas, N = 10000

In Figure 7 we provide the availability analyses for the results obtained in Figure 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, we provide a good foundation for data availability under variable circumstances.

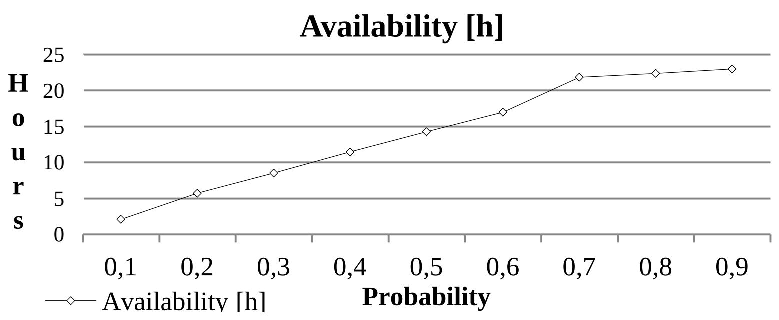

Figure 7 Average email availability/day

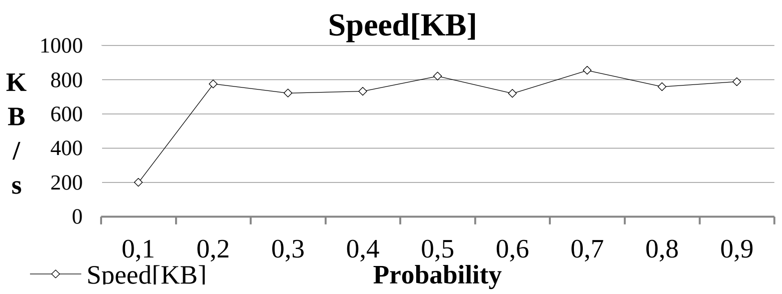

Figure 8 Download speed

Figure 8 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 we rely 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.

5.2 Conclusions

This report paper presented a new concept regarding the mailing infrastructure over a peer – to – peer network. We have shown a model of interconnecting peers according to the location services and dividing them according to the user behavior in time spent over the internet. Also we provided a thorough analysis 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.

In this report paper we have also solved the issues raised from the interoperability request between the P2P and client-server mailing architectures. We 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 paper pointed out the cases according to which our interface model represents a good solution for handling inter-compatibility between the two mentioned concepts.

As a future work we plan to extend our model over a self developed overlay concept, through which we can raise our expectations in terms of availability of email content over a period of time.

References

[1] O. Andy. *PEER TO PEER: Harnessing the benefits of Disruptive Technologies.* s.l. : O'Reilly Media, Feb 2001. ISBN:978-0-596-00110-0.

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

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

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

[5] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, S. Shenker. *Making Gnutella-like P2P Systems Scalable.* s.l. : SIGCOMM, 2003.

[6] I.Clarke, O.Sandberg,B.Wiley,andT.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, Junu 2000.

[7] ***.KazaA Homepage. http://www.kazaa.com.

[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 and Danny Bicson, academic supervisor Prof. Scott Kirkpatrick. *The eMule Protocol Specification.* January.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] D. Karger, E.Lehman, F.Leighton, M. Levine, D. Lewin, R. Panigrahy. *Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web.* s.l. : In Proceedings of the 29th Annual ACM Symposium on Theory of Computing.

[16] 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.

[17] 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.

[18] 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.

[19] Mayank Pandey,Syed Mushtaq Ahmed and Banshi Dhar Chaudhary. *2T-DHT: A Two Tier DHT for Implementing Publish/Subscribe.* s.l. : International Conference on Computational Science and Engineering, 2009.

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

[21] J. Kangasharju, K. W. Ross, 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

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

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

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

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

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

[27] ***.Wikipedia. http://en.wikipedia.org/wiki/Broadband_Internet_access

[28] G. Song, S. Kim, D. Seo, „Replica Placement Algorithm for Highly Available Peer-to-Peer Storage Systems“, First International Conference on Advances in P2P Systems, 2009.
[29] ***.http://tools.ietf.org/html/rfc5321
[30] P. Zimmermann. The Official PGP User’s Guide. The MIT Press, 1995.
[31] ***.http://www.google.com/about/datacenters/locations/index.htm