HMail: A hybrid mailing system based on the collaboration between traditional and Peer-to-Peer mailing architectures

Patrik Emanuel Mezo, Mircea Vladutiu and Lucian Prodan
Department of Computer Science
“Politehnica” University of Timisoara, Romania
patrik.mezo@yahoo.com, mircea.vladutiu@cs.upt.ro, lprodan@cs.upt.ro

Abstract—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.

This paper introduces a new mailing architecture design by combining our research activities on developing a hierarchical Peer-to-Peer (P2P) framework and resource evaluation methods of a certain peer in the network. We also group users 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.

Keywords: P2P mailing system, P2P mailing framework, P2P distributed mailing system.

I. INTRODUCTION

Nowadays one of the most common communication tools is electronic mail (e-mail or email). Built on a server-centric architecture, the mailing system relies on the client-server concept. An email client (mail user agent - MUA) 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. The mail transfer agent (MTA) describes how distributed tasks are handled among grouped entities (clusters), for assuring performance and quality of service in terms of data replication, location services, network availability or load balancing. The mechanism of store and forward usually describes the process of email operations between the described entities (MUA or MTA).

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 [1][2]. Another issue arises from the costs supported by the mail providers in terms of dedicated buildings distributed geographically (e.g. google.com datacenters [3]) and specialized trained personal 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 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 behavior (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.

Two different architectural concepts, hybrid and overlay, have attempted to solve this issue in P2P network implementations. The hybrid solution has promoted peers with above-than-average computing resources (Super Nodes - SN) over participants that could/would not share their resources. Mailing systems developed on this concept [4][5] were designed to rely their backbone on Super Node entities, establishing a reliable and stable network environment.

The overlay concept [6][7][8] 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 [9][10] are more complex in terms of architecture design, security, data availability and overall distribution of tasks.

Our mailing architecture solution relies on the framework provided by a hierarchical overlay framework design [11]. Our 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 we can specify on which computing resources the mailing operations can take place, and further, our solution can provide a better security of data regarding the email content, storing, sending and receiving operations. We chose to classify the hierarchical layers by the geographical criteria of peers because of the P2P network
behavior (instability of network). It is much faster to retrieve information from a peer located geographically closer to the requester.

II. BACKGROUND

Our P2P mailing infrastructure relies on a hierarchical extension of the Chord protocol. In this section we briefly describe both systems:

A. Chord

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 [7] is built in this manner, describing a ring-like virtual network (Fig. 1.a), 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 ith entry in the finger table at a node n contains the address of successor(n+2^i-1). Fig. 1.a 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.

B. Hierarchical extension

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 [11] 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).

In Fig. 1.b, the manner of building one hierarchical module is shown. Through linking the nodes with the dashed lines the first hierarchical layer is built. The dotted lines form the second layer only from linking nodes from the first layer. For the ease of representation, we will use the description found in Fig. 1.c, where we can define operations only at this level of representation. Each layer houses several partitions, formed through linking nodes from the layers situated below.

III. ARCHITECTURE PRELIMINARIES

There are two major structural components that build our 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 behavior 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 our previous research [4], we 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 we will appeal to the research found in [4] for describing the terms of joining one hierarchical module of the overlay.

A. Metrics Preliminaries

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 our P2P mailing architecture design we 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 requirement 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. We will use the algorithm found in [4], 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, our requirement expectation regarding this resource is focused more on the upload speed. We group nodes according to this metric because we want to provide quality of service regarding the email operations for our architecture design (sending/receiving).

The computing power designates the quantity of operations a certain peer can handle. Through quantity, we refer to the number of processes 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.

B. Basic Components

Our platform framework design relies on several hierarchical modules built above the Chord overlay (Fig. 2). Every HM identifies itself through the geographical area distribution of its nodes (peers). For this purpose we require the service of an additional application, such as MaxMind [12], which provides us with 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 we require that all user IDs should append the country code next to the domain name (ex. user_id@domain.country_code). 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 [13] 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. For the rest of this paper we will refer directly to the hierarchical layers of a certain HM through their partition definition for a better understanding.
IV. 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 our security model by requiring the services of an external certificate authority, which could provide a higher level of security.

Our P2P 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 [14].

We have identified three types of email operations on our mailing infrastructure: sending, receiving and deleting of email content. In these three basic categories we will also specify details regarding the store, authentication and garbage collector operations.

A. Send Operation

In Fig. 2 Alice’s computer represents one of the member peers of our mailing architecture design. Alice uses her common email client to write an email to Bob. 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. We assume that Alice’s computing machine is not validated by an HM. Therefore 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 we assume 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).

B. Receiving Operation

When Bob wants to check his inbox, he performs this operation through the MUA application. The request is sent through the email client by connecting locally to our 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.

C. Delete Operation

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.

V. INTEROPERABILITY SOLUTIONS

We have designed our mailing architecture according to the interface solution found in [15]. 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). We have considered two scenarios in handling operations with the traditional mailing service: standalone and hybrid mailing service.

When considered as a standalone P2P mailing application, our solution design must handle only incoming mails from the traditional architecture. For this purpose we need a number of few nodes from the spool area of certain defined HMs registered as MX-Hosts in the domain name system (DNS) [16] named after our 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 our 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 our 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 our 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 we are focusing 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 our 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.

VI. SIMULATION AND EXPERIMENTAL RESULTS

We have simulated our mailing architecture design in an object oriented environment, where peers have been represented as objects within the application. We simulated every participant to the network 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 [4]. Because peers can join or leave the network at anytime, we 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 Fig. 3 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 [4][10]. 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, our 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.

In Fig. 4, we have represented the data availability expectation for each considered uptime case scenario and we compared our results with [4]. 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 our simulation environment, our 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 rejoin 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.

VII. CONCLUSIONS

In this paper we have designed a new mailing architecture solution based on the P2P network model. We used our early research solutions for providing a
stable environment for the mail operations. We have shown two types of interoperability with the traditional mailing service, which proves that our mailing architecture design can handle well any type of requests from other mail services implementations.

We provided a thorough analysis regarding the simulation environment built for this purpose. We have shown that our mailing architecture design can be sustained from nodes that range between several uptime expectations and provide a stable data holder for each of the considered case scenarios.

ACKNOWLEDGEMENT

This work was partially supported by: 1.The strategic grant POSDRU/88/1.5/S/50783 (2009) of the Ministry of Labor, Family and Social Protection, Romania, co-financed by the European Social Fund – Investing in people; 2. The grant POSDRU / 21/ 1.5 / G /13798, inside POSDRU Romania 2007 – 2013 co-financed by the European Social Fund – Investing in People.

REFERENCES